tanulmányok 96/1979

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

KONTURKERESÉS ZAJOS DIGITALIZÁLT KÉPEKBEN

Kandidátusi értekezés

Irta:

MÉRŐ LÁSZLÓ

Tanulmányok 96/1979.

A kiadásért felel:
DR VÁMOS TIBOR

ISBN 963 311 087 4

ISSN 0324-2951

Készült a
SZÁMOK Reprográfiai Üzemében 9281

TARTALOM

***

0. Bevezetés

Ez a dolgozat az MTA SZTAKI-ban folyó intelligens robot kutatás keretében készült. A kutatás célja egy olyan szem-kéz rendszer kifejlesztése, amely ipari tárgyak felismerésére és szerelésére alkalmas. A dolgozat a tárgyak felismerésének első lépéseit mutatja be.

A dolgozatban tárgyalt feladat megfogalmazásához definiálnunk kell a látványgráf fogalmát.

Definició: A látványgráf egy sikbeli rajz. A látványgráf csucsai a sik pontjai, élei pedig egyenesszakaszok, vagy körivek a csucsok között.

A látványgráf tehát egyfelől egy sikbarajzolható gráf, amelynek éleit kétfajta cimkével /egyenes vagy köriv/ láttuk el, másfelől a látványgráfból a rajz siktopológiai tulajdonságai is kiolvashatók.

A tárgyak felismeréséhez első részcélul a következő feladatot tüztük ki: Állitsunk elő egy olyan látványgráfot, amelynek élei optimálisan közelitik az input képen látható tárgyak lapjainak határvonalait, csucsai pedig a tárgyak csucsait. Ennek a látványgráfnak az éleit konturvonalaknak nevezzük.

A dolgozat a látványgráf előállitásához szükséges algoritmusokat 3 fejezetbe csoportositva tárgyalja. Az egyes fejezetek bevezetőjében /a 0. pontokban/ exponáljuk a vizsgálandó részfeladatokat, és áttekintjük az idevonatkozó irodalomban fellelhető főbb eredményeket.

Az 1. Fejezetben egy uj élkereső algoritmust mutatunk be. Az élkeresésben négyzetes ablakokat használunk a képben. Képezzünk az ablak átlós kettéosztásával két lépcsősfüggvényt. Az ezekkel kapcsolatos konvolueiók hányadosa az ablakot átszelő éldarab iránytangensét adja. /1.1. pont. Állitás/. Ebből a tényből kiindulva dolgoztuk ki az élkereső algoritmust.

A 2. Fejezetben egy "naiv" konturkereső algoritmust irunk le, amely a látványgráf előállitásának problémáját a lehető legegyszerübb uton kezeli. Egyszerübb és kevéssé zajos képek esetén ez az algoritmus is kielégitő eredményt ad, de erősen zajos képek /főleg TV-képek/ esetén erősen megmutatkozik az algoritmus "naivsága", a matematikai modell hiánya.

A 3. Fejezetben először egy uj csomópontkereső eljárást mutatunk be, majd a csomópontok közötti optimális élsorozatokra egy matematikai modellt adunk meg /3.2. pont/. Ezután leirunk egy algoritmust, amely a matematikai modellt kielégitő élsorozatokat állit elő, és ebből kapjuk meg a látványgráfot.

A 4. Fejezet az algoritmusok számitógépes implementálását, és a kisérleti eredményeket mutatja be. Ez a fejezet a dolgozatnak az előzőekkel egyenértéküen fontos része, hiszen a használt matematikai modellek és algoritmusok adekvát voltát csak a kisérleti tapasztalatok bizonyithatják.

Ha az olvasó csupán vázlatos áttekintést akar nyerni az él- és konturkereső eljárásokról, elegendő a nulladik pontokat elolvasnia.

A vizuális input /szem/ ügyében párhuzamosan kisérletezünk lézeres és televiziós eszközzel. A TV készülék [18], vagy egy 144 x 192 pontból álló képmátrixot tud 16 szürkeségi szinttel közvetlenül az R-10-es számitógépbe bevinni, vagy egy 288 x 384 pontos felbontásu képet oly módon, hogy minden sorban csak a szürkeségi szint változások koordinátáit és a megváltozott szürkeségi szintet adja meg. A két képtárolási mód információtartalom szempontjából ekvivalens, de egy adott feldolgozási algoritmus hatékonysága a két esetben különböző lehet.

Más a helyzet a lézerkép esetében. A szerkezet müködési elve a következő: egy számitógéppel vezérelt akusztooptikai deflektor által kibocsátott fény visszaverődésének intenzitását mérik különböző helyeken elhelyezett fotoszenzitiv elemek, amelyek az intenzitás bizonyos küszöb fölötti változását jelzik. A kibocsátott fény végigtapogatja /szkenneli/ a szinteret. A visszavert fény

intenzitása ott változik, ahol a fény a szintéren lévő tárgyak élein áthalad, igy a kapott kép a szintéren látható éleket ábrázolja. Ennek a képnek már más az információtartalma, mint a TV-képnek, hiszen a szürkeségi szintekről nem mond semmit, csak az élpontokat tartalmazza, viszont sok hibával /zajosan/.

Mivel csak egyirányu letapogatás esetén a szkennelés irányával azonos, vagy ahhoz közeli irányu élek nem, vagy nagyon zajosan jelennek meg, két egymás utáni, egymásra merőleges irányu /pl. vizszintes, ill. függőleges/ szkennelésre van szükség.

A lézerszem legnagyobb előnye az, hogy csak a tárgyak éleit detektálja, függetlenül a megvilágitás körülményeitől. Hátránya viszont az ára, valamint az, hogy nehezen mozditható és zoomozható.

Feladatunk az, hogy a most leirt input eszközök által szolgáltatott digitalizált és zajos képekből állitsuk elő a látványgráfot. A felismerés további lépéseiben az igy kapott látványgráfban heurisztikus gráfelméleti és matematikai nyelvészeti módszerekkel [14, 15, 16, 27, 28] az eleve adott tárgyakról nyert elméleti látványgráfokkal azonos strukturákat keressünk, és igy azonositjuk a szintéren látható tárgyakat.

E helyen szeretném megköszönni Vámos Tibornak, csoportunk vezetőjének, és csoportunk tagjainak: Kovács Erikának, Galló Valentinának, Báthor Miklósnak és Siegler Andrásnak a sokrétü segitséget, ötleteket és biztatást, amit a dolgozat elkészitéséhez kaptam. Külön szeretném kiemelni Vassy Zoltán közremüködését, aki elinditott a téma irodalmának tanulmányozásában és sok hasznos gondolattal segitett. Köszönöm Csibi Sándor professzornak a segitségét és szakmai tanácsait, amelyek nagy mértékben hozzájárultak ahhoz, hogy ez a dolgozat jelen formájában elkészüljön.

1. Élek és rövid egyenesdarabok detektálása

1.0. Bevezetés

A kép fogalma többféle matematikai reprezentációt tesz lehetővé. Leginkább az elméleti ihletésü képfüggvény és a számitástechnikai ihletésü képmátrix fogalmát szokták használni.

A képfüggvény egy kétváltozós f(x,y) valós függvény, amelynek értelmezési tartománya az X-Y sik egy D tartománya /pl. az egységnégyzet/ és értéke D minden x, y pontjában egy valós szám, a fényesség értéke abban a pontban, azaz a szürkeségi szint.

A képmátrix egy mátrix, amelynek szintén minden eleme egy szürkeségi szint. Feltehetjük, hogy a mátrix minden eleme természetes szám, ugyanis a képmátrixot azonositani szokták a digitalizált input képpel. Ugy fogjuk fel, hogy a képmátrix a képfüggvény egy véges közelitése.

Feladatunk ezután felfogható ugy is, mint egy információredukciós feladat. Egy 142 x 192-es képmátrix minden pontban 16 szürkeségi szinttel közel 200 000 bit információt tartalmazna abban az esetben, ha az egyes pontok, mint valószinüségi változók függetlenek, és a lehetséges szinteket tekintve egyenletes eloszlásuak lennének. Egy értelmes /valamit ábrázoló/ kép esetében azonban ez messze nem áll fenn, sőt ellenkezőleg: nagyon is szoros a függőség a pontok között. A szintérnek egy nagy pontosságig megfelelő leirása is alig pár tucat bitet igényel /mely tárgyak vannak jelen az adott választékból, milyen állásban és hol/. Egy ilyen leirás értéke azonban már nem információtartalmában van, hanem abban, hogy a szintérről nyujt lényeges tájékoztatást, feltéve, hogy az egyes tárgyak /geometriai, topológiai, stb/ strukturáját eleve ismertnek vehetjük. Ez indokolja, hogy a kép felismerésénél első célnak az input képből a látványgráf előállitását tüztük ki.

Az emlitett információredukció első lépéseként a képben rövid egyenesdarabokat /éleket/ keresünk, amik a kép kis darabján /"ablakokban"/ optimálisan közelitik a konturvonalakat. Ebben a fejezetben ennek a feladatnak a megoldásával foglalkozunk.

Vegyük észre, hogy a lézerkép nem tesz eleget a képmátrix definiciójának, mert a mátrixban az értékek nem szürkeségi szintet jelentenek. Az ilyen, csak az élek helyét tartalmazó képmátrixot gradiens képnek szokták nevezni.

Számos módszert dolgoztak ki arra, hogy a képmátrixból jó minőségü /azaz kevéssé zajos/ gradiens képet nyerjenek. A következőkben bemutatjuk néhány ilyen eljárás alapgondolatát, majd a fejezet további részeiben olyan algoritmusokat ismertetünk, amelyek az input képből közvetlenül az egyenesdarabokat állitják elő. Az utóbbiak előnye, hogy lényegesen gyorsabbak, mert nem kell az input kép minden pontjára elvégezni egy élpontdetektáló algoritmust.

A legkézenfekvőbb módszer a gradiens kép előállitására kiszámitani magának a képfüggvény gradiensének az értékét, hiszen él ott van, ahol a gradiens értéke nagy. Roberts [61] a gradiens nagyságának közelitésére a következő egyszerü és logikus formulát használta: Jelölje g(i,j) a képmátrix (i,j)-edik elemét, és legyen:

$$\|\nabla g(i,j)\| \approx \left([g(i,j) - g(i+1, j+1)]^2 + [g(i,j+1) - g(i+1,j)]^2 \right)^{1/2} = R(i,j)$$

Szokás R(i,j) helyett a következő formulát is használni:

$$F(i,j) = |g(i,j) - g(i+1, j+1)| + |g(i, j+1) - g(i+1, j)|$$

F(i,j) számitása sokkal egyszerübb, és meglehetősen hasonlóan viselkedik, mint R(i,j). Egyébként látható, hogy

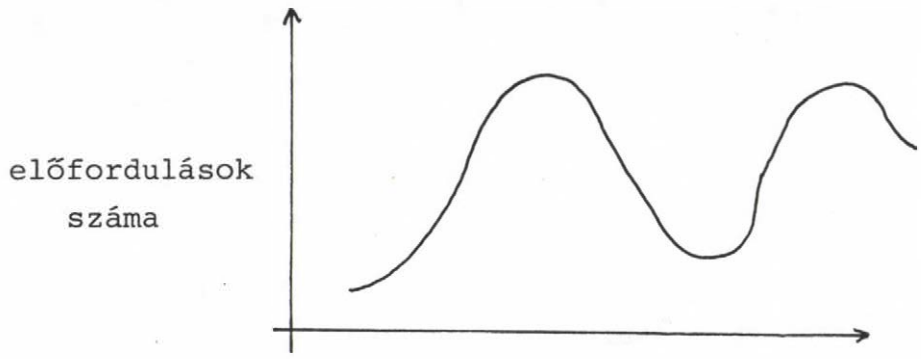
$$R(i,j) < F(i,j) < R(i,j)\sqrt{2} \quad .$$

A gradiensnek ez a számitása nagyon zajérzékeny. Ezért pl. Sobel [76] a pontok 3x3-s, Persoon [56] pedig egyenesen 6x6-os környezetében ad meg formulákat a gradiens nagyságának közelitésére.

Bonyolultabb differenciál-közelitő módszereket irnak le Kasvand [40], Wechsler és Kidode [81], Frei és Chen [26], Pingle [57], valamint Brooks [20].

Duda és Hart [4] a Fourier-analizis segitségével irnak le módszereket az élpontok detektálására. Eljárásuk alapeszméje az, hogy a tényleges, a szintér tárgyairól származó éleket felfoghatjuk magas frekvenciáju zajoknak, mig az igazi zajok alacsony frekvenciájuak. Igy alkalmas szürőfüggvénnyel az alacsony frekvenciáju zajok kiszürhetők, a magas frekvenciájuak pedig detektálhatók.

Egészen más hozzáállást választott az éldetektálás problémájához Chow [21]. Abból indult ki, hogy ha valahol a képen él van, akkor ott a környéken véve fel az ablakot, a szürkeségi szintek eloszlása bimodális lesz, hiszen az egyes szürkeségi szintek előfordulásainak száma az él két oldalára jellemző két szürkeségi érték körül csucsosodik ki /2.1.1. ábra/. Ezért igyekszik megtalálni a két csucs között a völgy mélypontját és az ablakokban mindenütt élet detektál ott, ahol a szürkeségi szintek átlépik ezt a küszöbértéket.



2.1.1. ábra

Ezt a küszöbértéket Chow azzal a feltevéssel keresi meg, hogy az adott ablakban az f képfüggvény $f_a(x)$ szürkeségi szint-eloszlása két normális eloszlás keverésével adódott, amelyek várható értéke, ill. szórása μ_2,σ_2 ill. μ_1,σ_1.

Legyen az él által elválasztott két terület aránya $\beta_1 : \beta_2$ ($\beta_1 + \beta_2 = 1$), ekkor belátható, hogy

$$f_a(x) = \sum_{k=1}^{2} \frac{\beta_k}{\sigma_k (2\pi)^{1/2}} e^{-(x-\mu_k)^2/2\sigma_k^2}$$

Ennek alapján meghatározható az $f_a(x)$ fenti előállitásában szereplő 6 paraméter optimális értéke ugy, hogy az ilyen paraméterekkel kapott $f_a(x)$ függvény L_2 - normában minimálisan térjen el a képfüggvényből kapottól /azaz a négyzetes hibát minimalizálja/. Ezután definiál egy "hegy-völgy-arányt", ami azt fejezi ki, hogy tényleg eléggé bimodálisnak tekinthető-e az eloszlás, és ha ez egy küszöböt nem halad meg, akkor abban az ablakban egyáltalán nem detektál élet.

Igen szellemesen alkalmazta a most bemutatott módszert Yachida és Tsuji [83] konturvonalak követésére. Hasonló elven müködő éldetektáló algoritmust dolgozott ki Weszka, Nagel és Rosenfeld [82].

Ismét egészen másképp fogják meg a problémát Montanari [49] és Martinelli [43] heurisztikus gráfelméleti élkereső algoritmusai.

Hozzáállásuk lényege a következő: Fogjuk fel a képmátrix minden elemét egy gráf csucspontjainak, mindegyik a mellette, alatta és fölötte lévő pontokkal legyen összekötve. A gráf minden éléhez egy költségérték is van rendelve, méghozzá az él által összekötött két csucs szürkeségi értékei különbségének abszolut értéke, levonva a maximális szürkeségi szintből. Ez azt jelenti, hogy a gráfnak egy éle annál "olcsóbb", minél inkább változik ott a szürkeségi szint, azaz minél inkább éle az eredeti képnek. Tehát két adott pont között a minimális költségü ut valószinüleg a kép élei mentén fog haladni, méghozzá fölösleges kitérők nélkül, azaz rögtön "zajt is szürve". A minimális költségü ut meg-

keresésének problémáját Martelli heurisztikus gráfelméleti algoritmussal oldja meg, Montanari pedig dinamikus programozással ugy, hogy az optimális ut hosszára eleve igyekszik egy ésszerü korlátot találni.

Ezzel a módszerrel igen jó minőségü gradiens képeket lehet előállitani. A módszer alkalmazásának fő problémája, hogy az optimális ut megtalálásához alkalmas peremfeltételeket kell biztositani. Vagy kezdő- és végpontot kell valahogyan találni, vagy pl. csak egy ablakra alkalmazni az eljárást, ahol eleve tudni, hogy az él az ablak tetejétől az aljáig megy. Nem biztos azonban, hogy két ablakban külön-külön talált optimális ut a két ablak egyesitésében is optimális utat alkot.

A gradiens kép keresésénél általánosabb feladat az, hogy a kép egy adott darabján /egy "ablakban"/ keressünk egy, az ablakban átmenő konturvonalat legjobban közelitő egyenesdarabot. Az ilyen egyenesdarabok többet mondanak a képről, mint a gradiens kép, hiszen mindenütt a konturvonal lokális irányát is közelitik. /A továbbiakban ezeket az egyenesdarabokat is éleknek fogjuk nevezni./

Ilyen egyenesdarabok keresésére a legkézenfekvőbb módszer az un. template matching /mintaillesztés/. Ahelyett, hogy meghatároznánk a keresett éldarab pontos helyét az ablakban, megtehetjük azt, hogy felvesszük néhány, éppen az adott ablakban centrált optimális él képét, és megvizsgáljuk, hogy az ablakban talált, f függvény melyikhez hasonlit legjobban. A hasonlóságot mérhetjük az l_1- vagy l_2-beli távolsággal. Az eleve kiválasztott optimális élmintákat nevezzük template-oknak. Holdermann és Kazmierczak [34] például 8 ilyen mintát használ az élek kiválasztására.

Rosenfeld és Thruston [67] széleskörü empirikus vizsgálatot folytatott a különböző template módszerek hatékonyságának megismerésére. A mintakereső módszerek élkeresésre való alkalmazásának fő akadálya az, hogy ha a kép nem pontosan valamelyik mintát tartalmazza, igen nagy a tévedés valószinüsége, sőt gyakran előfordul, hogy egy-egy, nem az ablak közepén átmenő élet teljesen kihagyunk. A másik baj az, hogy a próbálkozások nagy részének e-

redménye negativ, hiszen a sok élmintából csak egy az, amit keresünk. Ez nagymértékben lelassitja az ilyen algoritmusokat.

Nagyon szellemes és széles körben használt élkereső eljárást dolgozott ki Hueckel [36]. Eljárásának lényege, hogy az ablakban a képfüggvény Fourier-együtthatóinak segitségével kiszámitja a képet négyzetes hiba szerint legjobban közelitő egyenes paramétereit. Hueckel algoritmusát a függelékben részletesen ismertetjük.

A Hueckel-operátor egy megfelelőjét dolgozta ki O'Gorman [53]. A Fourier bázis helyett a Walsh-féle függvénysort használta bázisnak. Ez az eljárás kb. feleannyi gépidőt vesz igénybe, mint az eredeti Hueckel-operátor.

Végül emlitsük meg a Griffith [29] által kidolgozott, matematikai statisztikai ihletésü élkereső eljárást. A módszer alapeszméje az, hogy egy pont körül elvileg minden lehetséges idealizált élfüggvényre /és gyakorlatilag is nagyon sokra/ kiszámitja, hogy mi a valószinüsége annak, hogy - normális eloszlásu zajt feltételezve - épp az adott input képet kapja az illető élfüggvényből. Ezután ezek segitségével kiszámolja, hogy mi a valószinüsége egyáltalán a szóbanforgó ponton átmenő él jelenlétének az adott input kép esetén. Griffith szép eredménye, hogy sikerült erre a valószinüségre egy, csak a pont környezetében lévő szürkeségi szintektől függő, zárt képletet kapni. Ezek után élet detektál azokban a pontokban, ahol az él jelenlétének valószinüsége meghalad egy küszöbértéket.

Griffith algoritmusával nagyon szép éleket kapott poliéderekből álló szinterekben, de az eljárás, még számos közelitő lépés után is, nagyon sok számitást igényel.

Érdekes visszfényt vetnek az eddig elmondottakra Hubel és Wiesel [35] neurofiziológiai kutatásai. Vizsgálataik azt mutatták ki, hogy az emlősök szemének retinasejtjei elvégzik a rövid egyenesdarabok, élek kiemelését, még mielőtt a vizuális információ egyáltalán eljutna az agyba és oda már maguk az élek továbbitódnak.

Noha az emberi agy müködésének algoritmusairól vajmi keveset tudunk, egy számitógépes algoritmusnak külön érdekessége lehet, ha nincs ellentétben az emberi agyról szóló neurofiziológiai ismeretekkel, tehát nincs kizárva, hogy az agy megfelelő funkciójának modellezésére is alkalmas. Természetes számunkra jelenleg az algoritmusok számitógépes hatékonysága mérvadó.

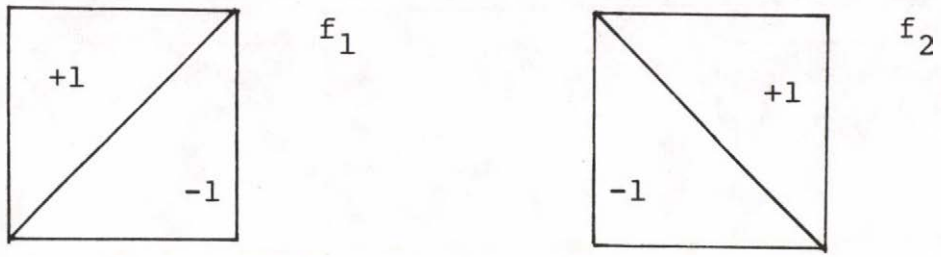
Végül idézzük ezzel kapcsolatban Hueckel [36] megjegyzését: "Lehet, hogy az élfelismerés problémája kisiklik az emberi tudatosság határai közül: az, hogy az élek eleve 'ott lévőknek' látszanak, hajlamossá tesz minket arra, hogy a probléma bonyolultságát alábecsüljük."

1.1. Egy uj élkereső algoritmus

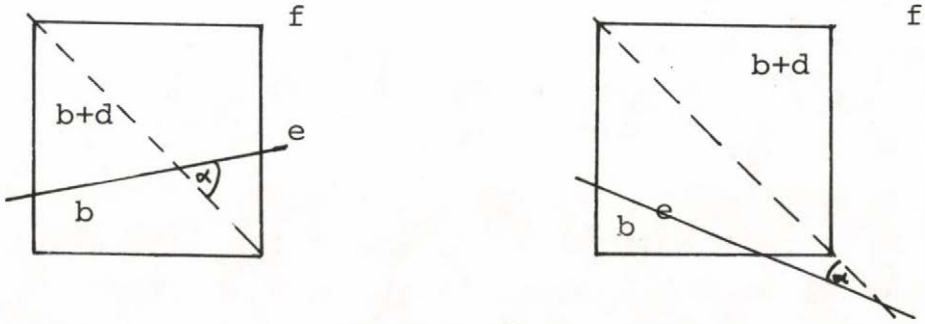
Hueckel eredeti interpretációjában a cél az volt, hogy az input-képben a legjobban illeszkedő egyenes-függvényt /éldarabot/ megtalálja. A következő állitás azt mutatja, hogy már két mintafüggvény segitségével is egy Hueckel-féle értelemben vett, optimális él irányáról mindent megtudhatunk.

Állitás: Legyen I az egységnégyzet, az f_1 függvény legyen +1 I mellékátlója fölött és -1 alatta, az f_2 függvény pedig legyen +1 az I főátlója fölött és -1 alatta. /1.1.1. ábra/. Legyen továbbá f egy olyan függvény, amelynek értéke I-n b+d egy I-t metsző e egyenes fölött és b alatta. /Két jellegzetes eset az 1.1.2. ábrán látható./ Jelölje α az e egyenes és I mellékátlójának a szögét /az 1.1.2 ábra szerint definiálva/. Ekkor

$$\frac{\int_I f \cdot f_1 \, dx\, dy}{\int_I f \cdot f_2 \, dx\, dy} = \operatorname{tg} \alpha$$



1.1.1. ábra



1.1.2. ábra

Bizonyitás: Szimmetria-okokból feltehetjük, hogy $45^\circ < \alpha < 90^\circ$. Először tegyük fel, hogy e áthalad I jobb felső csucsán /1.1.3. ábra/. Ebben az esetben az ábra jelöléseit, valamint f, f_1 és f_2 definicióját használva, ha T_1 az ACD_Δ és T_2 az ABD_Δ területe,

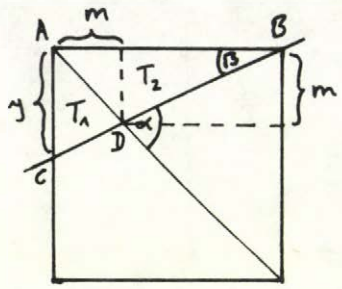
$$\frac{\int_I f \cdot f_1 \, dx\, dy}{\int_I f \cdot f_2 \, dx\, dy} = \frac{T_2 + T_1}{T_2 - T_1}$$

Mivel I egységnégyzet,

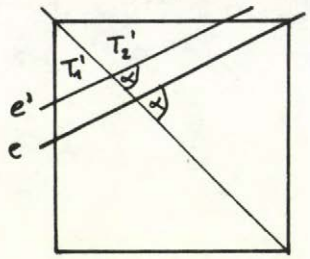
$$y = \operatorname{tg}\beta = \operatorname{tg}(\alpha - 45^\circ) = \frac{\operatorname{tg}\alpha - 1}{1 + \operatorname{tg}\alpha}$$

igy

$$\frac{T_2 + T_1}{T_2 - T_1} = \frac{\frac{(1+y)m}{2}}{\frac{(1-y)m}{2}} = \frac{1+y}{1-y} = \operatorname{tg}\alpha$$



1.1.3. ábra



1.1.4. ábra

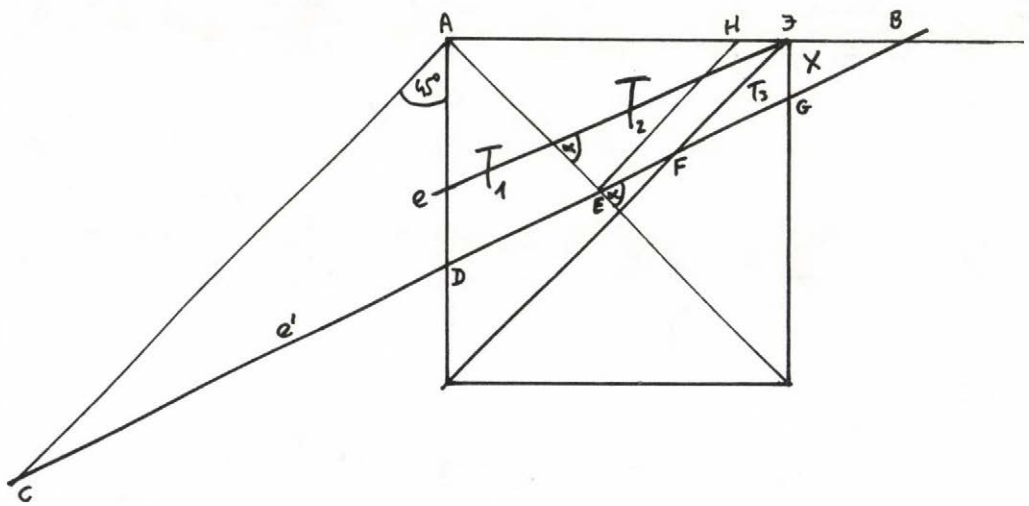
Most tegyük fel, hogy az e' egyenes az előbb vizsgált e egyenes fölött van, és vele párhuzamos /1.1.4. ábra/.

Ekkor a párhuzamosság miatt

$$T_1' = kT_1 \quad , \quad T_2' = kT_2$$

$$\frac{T_2' + T_1'}{T_2' - T_1'} = \frac{T_2 + T_1}{T_2 - T_1} = tg\,\alpha$$

Harmadszor, tegyük fel, hogy e' szintén e-vel párhuzamos, de e alatt fekszik, másrészt viszont I középpontja fölött /1.1.5. ábra/.



1.1.5. ábra

Az 1.1.5. ábra jelöléseivel legyen

$$T_1 = T_{ADE_\Delta}, \quad T_2 = T_{AEFJ}, \quad T_3 = T_{FGJ_\Delta}, \quad X = T_{JGB_\Delta}$$

és akkor, mivel e és e' párhuzamos, és a bizonyitás első része nemcsak egység-, hanem tetszőleges $|\overline{AB}|$ élhosszuságu négyzetre is igaz,

$$\frac{T_2 + T_3 + X + T_1}{T_2 + T_3 + X - T_1} = \operatorname{tg} \alpha$$

Tehát csak azt kell bebizonyitani, hogy

$$\left(\operatorname{tg}\alpha = \right) \frac{T_2 + T_3 + X + T_1}{T_2 + T_3 + X - T_1} = \frac{T_1 + T_2 - T_3}{T_2 + T_3 - T_1} \left(= \frac{\int_I f \cdot f_1 \, dx \, dy}{\int_I f \cdot f_2 \, dx \, dy} \right)$$

Vonjunk le az egyenlőség mindkét oldalából 1-t, vegyük a reciprokát és adjunk mindkét oldalhoz 1-et, akkor azt kapjuk, hogy

$$\frac{T_2 + T_3 + X}{T_1} = \frac{T_2}{T_1 - T_3}$$

azaz

$$\frac{T_1 - T_3}{T_1} = \frac{T_2}{T_2 + T_3 - X}$$

ismét 1-et levonva, átszorzás után kapjuk, hogy

$$\frac{T_1}{T_2 + T_3 + X} = \frac{T_3}{T_3 + X}$$

Ehhez viszont már csak azt kell megmutatni, hogy

$$\frac{DE}{EB} = \frac{FG}{FB}$$

Az FJB_{Δ} és a CAB_{Δ} hasonlósága miatt $\frac{FG}{FB} = \frac{CD}{CB}$, tehát csak annyi kell, hogy $\frac{DE}{EB} = \frac{CD}{CB}$, azaz $\frac{CD}{DE} = \frac{CB}{EB}$. Mivel AD a CAE szögfelezője, $\frac{CD}{DE} = \frac{CA}{AE}$ és a CAB_{Δ} és az EHB_{Δ} hasonlósága folytán $\frac{CB}{EB} = \frac{CA}{EH}$. Mivel az AEH_{Δ} egyenlőszáru, AE = EH , és ezzel ebben az esetben is bebizonyitottuk az állitást.

A bizonyitás során sehol sem használtuk fel, hogy d pozitiv, igy ha az egyenes I középpontja alatt van, vagy ha a két felén a szürkeségi szint-értékeket felcseréljük, a bizonyitás érvényes marad, tehát az állitást bebizonyitottuk. ■

A most bizonyitott állitás számunkra különösen fontos érdekessége, hogy a két integrál hányadosa nem függ sem az egyenes két oldalán a szürkeségi szintektől, sem az egyenes tényleges helyétől, csakis az iránytangensétől. Külön öröm számitástechnikai szempontból, hogy nem köralaku ablakban kell számolnunk /mint a Hueckel-operátornál/, hanem négyzet alakuban.

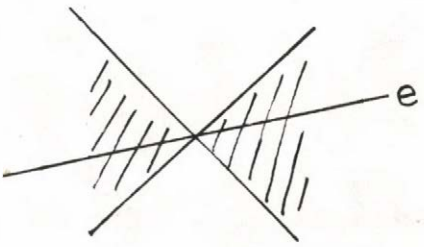
f_1 és f_2 felfogható ugy is, mint egy ortogonális függvénysorozat első két tagja, amely függvénysorozat azzal az érdekes tulajdonsággal rendelkezik, hogy a Hilbert-tér minden "egyenest ábrázoló" függvényének iránytangense már az ortonormált bázis első két tagjával vett Fourier-együtthatók alapján meghatározható.

A két integrál a képmátrixból nagyon gyorsan számitható, és mivel az integrálás nagy területeken történik, a módszer zajérzékenysége is igen csekély. I mérete az input kép zajosságától függően választható. Mi 5 x 5-ös és 9 x 9-es négyzet között változtattuk. /Minél nagyobb az ablak, annál pontosabbak lesznek az egyenesdarabok, viszont annál kevésbé tudunk finom részleteket megkülönböztetni a képben./

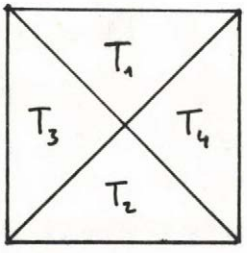
Következő feladatunk az, hogy ha már ismerjük az él iránytangensét, akkor találjuk meg az él pontos helyét is az ablakban. Mint az algoritmus végén majd kiderül, szimmetria-okokból egyenlőre feltehetjük, hogy az él az 1.1.6. ábrán mutatott siknegyedben van /azaz az iránytangens előjele pozitiv/.

Az él elhelyezésének módszere a következő:

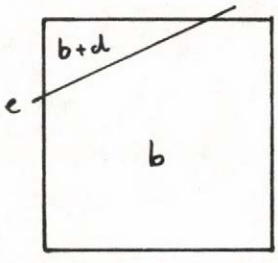
Megbecsüljük először is, hogy mennyi lehet az adott ablakban a b és a b+d érték, és az egész ablakban vett szürkeségi szintek összegéből kiszámitjuk, hogy ha az él vizszintes lenne, hol haladna, feltéve, hogy a kép alsó fele sötétebb, mint a felső. /Azaz a négyzetet alulról "meddig lehetne megtölteni" a b+d-kkel./ Ezután ezt a vizszintes élet a szükséges szöggel elforditjuk a középpontja körül. Ha az igy kapott él nem "lóg ki" a négyzetből, akkor készen vagyunk, ha pedig kilóg, akkor még annyira lejjebb kell tolni, hogy a b+d-s és b-s területek aránya ne változzon.



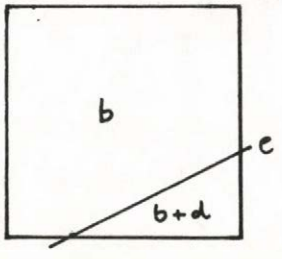
1.1.6. ábra



1.1.8. ábra



1.1.7/a. ábra



1.1.7/b. ábra

Ezután még meg kell állapitani, hogy a 1.1.7. ábra két esete közül melyik áll, azaz, hogy helyes volt-e az a feltevés, hogy a kép az él fölött világosabb, mint alatta. Most már látható, hogy ha az iránytangens negativ volt, akkor ezek után a kapott élet egyszerüen tükrözni kell a négyzet mellékátlójára.

Mivel viszonylag kis ablakokkal és digitalizált képekkel dolgozunk, az előbb vázolt algoritmust egyszerü közelitő lépésekkel hajtjuk végre. Először is az iránytangens abszolut értékéből kapott szög értékét ugy adjuk meg, hogy kiszámutjuk t értékét ugy, hogy ha az él egyik vége az n x n-es ablaknégyzet /0,0/ pontjában van, akkor a másik vége az ablak /t,n/ pontjában legyen. /$-n < t < n$/. t tehát azt fejezi ki, hogy ha az él két végpontjának x-koordinátája közötti különbség n, akkor az y-koordináták különbsége mennyi.

A b és d mennyiségeket az 1.1.8. ábrán látható területekből becsüljük /ezeket az összegeket az iránytangens kiszámitásához már ugyis megkaptuk/.

A következő közelitést használjuk:

$$b = \left[\frac{4\,(\min\{T_i\} + 2n)}{n^2} \right]$$

$$d = \left[\frac{4\,(\max\{T_i\} + 2n)}{n^2} \right]$$

ahol n x n-es az ablak, és $[\cdot]$ az egészrész-függvény jele.

Ezután az él bal ill. jobb végpontjának y-koordinátáit igy számoljuk /az x-koordináták egyenlőre 0 ill. n/: $y_1 = S' - \frac{t}{2}$, $y_2 = S' + \frac{t}{2}$

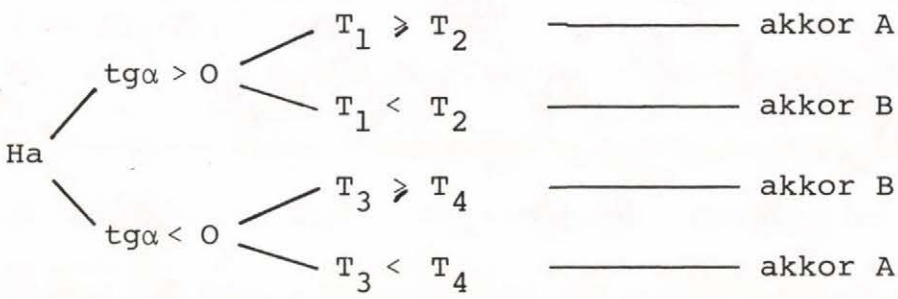
ahol $$S' = \frac{\int_I f\,dx\,dy - bn^2}{dn}$$

Ha y_1 vagy y_2 negativ, akkor egyszerü közelitéssel /ez már csak igen kis hibát okoz/ pl. $y_1 < 0$ esetén

$$y_1' = 0 \qquad y_2' = y_2 - \frac{y_1}{2} \qquad x_1' = \frac{y_1}{2} \qquad x_1' = n$$

végpont-koordinátákat számolunk. Ezután már csak azt kell eldönteni, hogy az 1.1.7. ábrán mutatott a/ vagy a b/ eset áll-e

fenn. Ezt igy döntjük el:



ahol A az él helybenhagyását, B pedig az I középpontjára való tükrözését jelenti. Ezután még ha tgα negativ volt, akkor az élet tükrözzük I főátlójára és megkaptuk a keresett élet az ablakban.

Algoritmusunk hibája, hogy minden körülmények között talál élet, akkor is, ha nincs és akkor is, ha nem egyenes él van az ablakban, hanem például a kontur épp ott törik, vagy több él fut össze.

Az első problémán könnyen segithetünk, hiszen természetesen nem kell, hogy élet keressünk, ha d nem ér el egy küszöböt. A küszöb állitásával kereshetünk finomabb vagy durvább éleket.

A második problémára az algoritmuson belül nem tudunk megoldást találni, de ha valahol nem igazi élet találtunk, az a környező egyenesdarabokból kiderül, igy ennek felderitését későbbi algoritmusainkra bizzuk.

Ha viszont van egyenes él az ablakban, azt algoritmusunk igen jó hatásfokkal és pontossággal mutatja ki: ebben nem marad el hatékonysága a Hueckel operátortól.

Az eljárás zajérzékenységére például bináris képek esetén egyszerü és durva számolással alsó becsléseket kaphatunk a következő uton: tegyük fel, hogy egy α szögü egyenes /$0 < \alpha \leq 45^{\circ}$/ ideálisan digitalizált változatában k pont képét megváltoztatjuk. Ekkor kiszámitható, hogy mi a valószinüsége annak, hogy

$$\alpha - \varepsilon < \operatorname{arctg} \frac{\int_I f_1 \cdot f' \, dx\,dy}{\int_I f_2 \cdot f' \, dx\,dy} < \alpha + \varepsilon$$

ahol f' a megváltoztatott kép, ε lehet pl. 10^{o}. A számolás azért végezhető el, mert a két integrált csak az befolyásolja, hogy a k pontból hány esik az 1.1.8. ábra T_1, T_2, T_3 ill. T_4 területére. Felhasználva, hogy

$$\frac{\Pi}{4} x < \text{arc tg}x < \frac{\Pi}{4} x + \frac{1}{4}$$

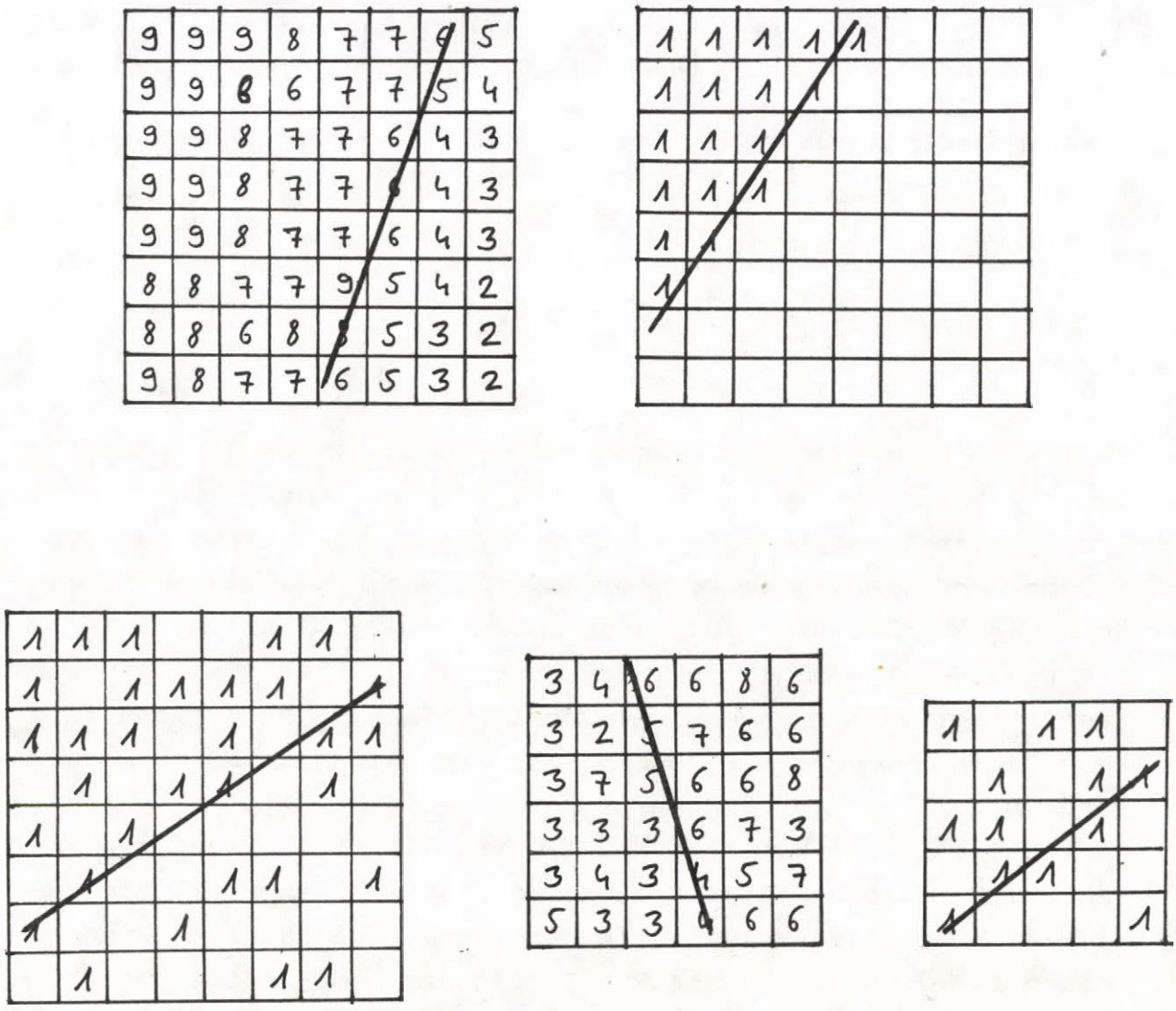
elég pl. 0 és 45^{o}-os egyenes esetén $\varepsilon = 7^{o}$-ra számolni, és belátható, hogy abból minden más esetre is következik az egyenlőtlenség $\varepsilon = 10^{o}$ -ra. A hányados érzékenysége a hibákra annál nagyobb, minél inkább az ablak szélén megy át az egyenes, igy fel kell tenni, hogy az egyenes által az ablakból lemetszett mindkét rész területe legalább az ablak területének 1/3-a. Ilyen körülmények között viszont már könnyen meghatározhatók a hibás pontoknak olyan eloszlásai, amelyeknél az okozott hiba biztosan ε alatt lesz. A hosszu, de egyszerü számolásokat nem részletezzük. k = 6 hibapontra egy 8 x 8-as ablakban pl. azt kapjuk, hogy minimum 94% valószinüséggel a hiba 10^{o} alatt lesz. Még k = 9 -re is 80% fölötti értéket kapunk.

Az input eszközökkel kapott képek azt mutatták, hogy a zajok eloszlása messze nem egyforma a képen: az élek körül inkább vannak zajok, mint a homogén régiókban. Igy a számitásoknál érdekesebbek, és többet mondanak azok a kisérletek, hogy egy zajos ablakban hová tenne az ember élet a szemével, és hová tesz az operátorunk. Az igy kapott eredmények sem rosszabbak, mint a számitásokból kapottak. Az algoritmus müködéséről nyert tapasztalatokat bővebben a 4. fejezetben ismertetjük.

Az 1.1.9. ábrán néhány példát mutatunk be, az első példa azonos Hueckel a függelékben idézett próbaábrájával, csak sarkokkal kiegészitve. A kapott él, mint látjuk, lényegében azonos. A bináris képekben az üres helyekre 0 értendő.

Az emlitett hiányosságokat az algoritmus sebessége ellensulyozza. Az algoritmus 250-350 gépi kódu utasitás végrehajtásával talál élet, és ez több, mint 16-szor kevesebb, mint amennyi a Hueckel-operátorhoz szükséges. Ráadásul lebegőpontos müvelete-

ket egyáltalán nem használ, és igy könnyen hardweresithető.



1.1.9. ábra

Jegyezzük még meg, hogy az él centráltságára az ablakban, /ha van él jelen/, könnyen kapható mérőszám az algoritmusból: S'-nek $\frac{n}{2}$ -től való eltérése épp ezt méri. Ha S' - $\frac{n}{2}$ = 0, az él

épp az ablak közepén megy át és $S' - \frac{n}{2}$ minél nagyobb, annál inkább az ablak szélén halad az él. Ebből az él "jóságára" kapott W mérőszámot

$$W = 1 - \left| S' - \frac{n}{2} \right| \cdot \frac{2}{n}$$

alakban definiáltuk, igy mindig $0 \leq W \leq 1$, és W annál nagyobb, minél "biztosabb" az él.

1.2. Egyenesdarabok a lézer-képben

A lézer-inputból kapott gradiens képre is ugyanazt a feladatot tüzzük először ki, mint a TV-ből kapott képmátrixra: kisebb ablakban optimális egyenes éldarab megtalálását. Ha ez megvan, akkor a felismerés további lépései már teljesen hasonlóan végezhetők, az input választásától függetlenül. Itt is csak azt szeretnénk elérni, hogy ha van egyenes éldarab a képben, ezt kapjuk meg; ha nem egyenes él van az ablakban, akkor most is a későbbi feldolgozásra bizzuk a hiba korrigálását. Fő szempontunk tehát most is a gyorsaság és az egyenes élek pontos előállitása.

Hueckel [37] kiterjesztette operátorát erre az esetre is: sikerült a megoldó tételét általánositania olyan alaku függvényekre, ahol a szürkeségi szint értéke két párhuzamos egyenes fölött b_1 köztük b_2, alattuk b_3 . Látható, hogy épp ilyesmi érdekel minket is. Vékony vonalak esetén viszont, /és minket elsősorban ez az eset érdekel/, Hueckel a jobb eredmény érdekében kénytelen volt a figyelembe vett bázisfüggvények számát megnövelni, ami tovább lassitotta az eljárást. Másrészt esetünkben amugyis csak az fordulhat elő, hogy $b_1 = b_3 = 0$, $b_2 = 1$, ami lényegesen leegyszerüsiti dolgunkat.

Ennek fényében fogalmazhatjuk ugy is a feladatot, hogy a négyzetben az adott pontokhoz illesszünk egy egyenest. Minima-

lizálandó mennyiségnek választhatjuk vagy a pontoknak az egyenestől való távolságaik négyzetösszegét, vagy valamelyik koordináta irányában vett távolságok négyzetösszegét. A második módszer előnye, hogy sokkal gyorsabban végrehajtható, hátránya viszont, hogy olyan pontokra, amelyek egy az adott koordinátával párhuzamos egyenes mentén helyezkednek el, nagyon hibás eredményt tud adni /ld. 1.2.1. ábra/. Ezt a nehézséget viszont áthidalja az a tény, hogy a lézer-inputunk amugyis kétszeres képet ad: egyet a vizszintes, egyet pedig függőleges letapogatással. Igy tehát ha a vizszintes letapogatásból kapott képnél a vizszintes koordináta szerinti távolságok négyzetét minimalizáljuk, a függőlegeseknél pedig a függőlegesekét, akkor teljesen elkerülhetjük ennek a jóval kevesebb számolást igénylő módszernek a hátrányait.

A pontoknak az egyenestől való távolságának négyzetösszegét minimalizáló /tehát: "korrektebb"/ egyenes a következőképpen kapható meg:

Legyenek az élpontok irányvektorai $\underline{V}_i = (x_i, y_i)$. Először is belátható, hogy az optimális egyenes átmegy az élpontok sulypontján. Ezután bebizonyitható, hogy az optimális egyenes iránya párhuzamos az

$$S = \sum_{i=1}^{n} \underline{V}_i \, \underline{V}_i^t$$

szimmetrikus mátrix legnagyobb sajátértékéhez tartozó sajátvektorral. Ebből már az optimális egyenes könnyen megkapható. Az emlitett eljárás bizonyitása megtalálható Duda és Hart [4] könyvében.

A másik, általunk is használt eljárás pedig a következőképpen müködik: Legyen (0,0) a bal alsó sarka az n x n -es ablakunknak, és

$$k_{ij} = \begin{cases} 1, \text{ ha van input jel az } (i,j) \text{ pontban} \\ 0, \text{ ha nincs.} \end{cases}$$

Azt az y = ax+b egyenest keressük, amely minimalizálja a

$$H = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} k_{ij} \left(ai + b - j\, k_{ij}\right)^2$$

kifejezést, azaz amelyre $\frac{\partial H}{\partial a} = \frac{\partial H}{\partial b} = 0$.

A gyors számolás érdekében használjuk a következő jelöléseket:

$$S_1 = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} k_{ij} \qquad S_2 = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} i\, k_{ij} \qquad S_3 = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} j\, k_{ij}$$

$$S_4 = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} ij\, k_{ij} \qquad S_5 = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} i^2 k_{ij}$$

Ekkor azt kapjuk, hogy

$$a = \frac{S_1 S_4 - S_2 S_3}{S_1 S_5 - S_2^2} \qquad b = \frac{S_3 S_5 - S_2 S_4}{S_1 S_5 - S_2^2}$$

Az optimális él kiszámitása igy körülbelül ugyanannyi ideig tart, mint a TV-kép esetén.

Mindkét emlitett legkisebb négyzetes módszer hátránya, hogy nagyon érzékeny egyedi, de nagyon kirivó zajokra. Az 1.2.2. ábrán látható pontokra például mindkét algoritmus az ábrán látható, nyilván nagyon hibás egyenest adja eredményül.

1.2.1. ábra

1.2.2. ábra

Ezen a problémán ugy próbálunk segiteni, hogy az ablakban előszöris megpróbáljuk elhagyni azokat a pontokat, amelyeknek nincs legalább két szomszédjuk az ablakban. /Mind a 8 szomszédos négyzetet figyelembe véve/. Ha az n x n -es ablakban ezután

még marad $\frac{n}{2} + 1$ pont, akkor csak ezekre végezzük el a legkisebb négyzetes illesztést. Ha tul kevés pont maradt igy meg, akkor először megpróbáljuk csak azokat a pontokat elhagyni, melyeknek nincs szomszédjuk, és ha igy is kevés pont marad, akkor mégiscsak az eredeti pontokkal dolgozunk.

Mindezek a bajok abból adódnak, hogy a legkisebb négyzetes illesztés egy hibás pontot annál nagyobb sullyal vesz, minél hibásabb. /Méghozzá a hibával négyzetes nő a suly./ Számunkra ez semmiképp sem szerencsés jelenség, mert esetünkben egy bizonyos hiba fölött már teljesen mindegy, hogy a zajpont hol van. Ezért szoktak olyanfajta optimalizálási kritériumokat is felvenni, hogy pl. legyen 1 a hiba, ha a távolság az egyenestől bizonyos előre adott ε-nál nagyobb, és 0, ha nem. Sajnos azonban ezek a kritériumok analitikusan teljesen kezelhetetlennek bizonyultak, és bár készültek ilyenfajta kritériumokat minimalizáló kereső algoritmusok, ezek általában igen lassuak, mert csak próbálgatással tudnak dolgozni.

2. A látványgráf előállitása

2.0. Bevezetés

Feladatunkat, a látványgráf felépitését igy viszonylag kevés szerző tüzte ki közbülső célnak. Többnyire csak azok tették ezt meg, akik a poliéderek világában terveztek felismerő algoritmusokat. Akik bonyolultabb szinterek /földi vagy holdbéli tájak, emberi arcok, ujjlenyomatok, biológiai képek/ felismerésével foglalkoznak, azok inkább jellegzetes egyenes vagy görbe vonalak keresnek a képben /mint pl. Shirai [72, 73] , vagy a homogén területek alakja alapján következtetnek /pl. Duda és Nitzan [24] /.

Az eddigi intelligens szem-kéz rendszerek tervezői vagy eleve tudták, hogy a tárgyak milyen nézetből látszanak és onnan milyen alakuak /mint pl. az edinboroughiak [18] , Ishii és Nagata [38] , vagy Uno, Ejiri és Tokunaga [38] /, és azután a templát módzsereket tudták használni, vagy csak arra használták a szemet, hogy helyesbitsék az egyébként vak, de intelligens kezet, ha esetleg mellényult /pl. Perkins és Binford [55] /. Ez utóbbi esetben egyszerü jellegzetességek megkeresésére egyszerüsödött a feladat, mint például bizonyosfajta speciális csucsok helyének meghatározása.

Mi nem mondunk le arról, hogy pontos tudomásunk legyen a felismert tárgyak milyenségéről /pl. épp nem látható részeikről is/. Ezért kerülünk hasonló vágányra azokkal a kutatókkal, akik a poliéderek világát akarják felfogni számitógéppel. Mig azonban ők a poliédereket, vagy azok egy osztályát általában igyekeznek megérteni, mi nem feltétlenül poliédereket ugyan, de konkrét tárgyakat akarunk leirni.

Roberts [61] a következő algoritmussal állitja elő a /persze csak egyenesekből álló/ látványgráfot: az 1.0. pontban emlitett élkereső eljárásával kapott gradiens kép pontjait addig füzi fel

egy egyenesre, amig a mindenkori utolsó öt pont egy egyenesre esőnek mondható. A pontokat akkor mondja egy egyenesre esőnek, ha eltérésük a rájuk fektetett optimális egyenestől egy korlát alatt van. Igy kap egyenesdarabokat. Az egyenesdarabok végpontjait, mindig a legközelebbieket, összehuzza. Ha két igy összehuzott egyenes vehető egy egyenesnek, egyesiti őket. Azokat a rövid egyenesdarabokat, amelyek nem illenek hosszabb egyenesekbe, eliminálja.

Roberts nagyon szép látványgráfokat kapott, algoritmusa azonban igen lassu, hiszen minden egymásutáni pontötösre ki kell számitani az illeszkedési kritériumot, és az egyenesdarabok összeillesztésénél is sokat próbálgat.

Sokkal hatékonyabb algoritmust használ Shirai [73], bár több megszoritást is tesz a tárgyakra. Csak konvex poliéderekből álló szinterekkel dolgozik és feltételezi, hogy a tárgyakat a háttértől eleve könnyen el lehet választani, pl. mert a háttér sötét.

Ezután egy, Roberts algoritmusához hasonló eljárással meghatározza a külső kontur egyeneseit. Shirai az egyenesek követésénél nemcsak az utolsó néhány pontot használja, mint Roberts, hanem minél hosszabb darabon talált jól egy egyenesre illeszkedő pontokat, annál "türelmesebb" a későbbi hibákkal szemben. A külső konturok alapján /tudva, hogy csak konvex poliéderekkel dolgozik/ javaslatokat tesz az egyeneskeresőnek, hogy hol sejthető belső él. A külső kontur konkáv csucsainál megkisérli egyenesen folytatni a külső éleket. Ha ez nem sikerül, akkor megpróbál egyéb belső élet keresni abból a csucsból kiindulva. Igy tovább, hierarchikusan egymás után következő próbálkozásokkal keresi meg a teljes konturt. Az eljárás előnye, hogy mivel mindig csak olyan éleket keres, amelyek biztosan elképzelhetőek, semmi esetre sem kaphat eredményül ellentmondásos látványgráfot. Bizonyos szinguláris esetekben azonban előfordulhat, hogy egy-egy belső élet nem kap meg, mert meg sem próbálja keresni.

Hasonló gondolatokat használ Griffith [30] is, bár keresési

stratégiája lényegesen különbözik Shirai módszerétől.

A mi feladatunk az eddigieknél valamivel egyszerübb, nekünk ugyanis egyenesdarabokkal kell dolgoznunk és nem egy gradiens kép egyes pontjaival. Egyenesdarabokból hosszabb egyenesek összeállitására igen szellemes módszert ir le Duda és Hart [23], a Hough-transzformáció [4] egy változatának alkalmazásával. Módszerük ötlete az, hogy az X, Y sikban lévő mindegyik egyenesdarabhoz határozzuk meg a ϱ_i, ϑ_i paramétereket ugy, hogy az egyenes egyenlete éppen

$$\varrho_i = x \cos \vartheta_i + y \sin \vartheta_i$$

legyen és minden egyenesdarabot feleltessünk meg a ϑ, ϱ sik ϑ_i, ϱ_i pontjának. Ekkor kolineáris egyenesdarabok a ϑ, ϱ siknak ugyanabba a pontjába fognak leképeződni. Mivel az egyenesdarabok nem pontosan kolineárisak, valójában az egymáshoz közeli pontok fognak egy hosszabb egyenesnek megfelelni. Most tehát egy cluster-analizist kell végrehajtani a ϑ, ϱ térben az egyenesdaraboknak megfelelő pontokon, és igy összeállithatjuk a kontur hosszabb egyeneseit.

Természetesen az egyenesek végpontjait az egyenesdarabokból kell meghatározni, vigyázva arra, hogy az ábrán lehet több, nem csatlakozó kolineáris egyenes, és azokat is külön kell választani.

A Hough-transzformáció módszert Shapiro [70] többféle görbe azonositására is kidolgozta, és vizsgálta az eljárás pontosságát zajos képekre.

A továbbiakban ebben a fejezetben leirunk egy igen gyors direkt algoritmust az egyenesdarabok egymásutáni generálására és a csomópontok detektálására. Algoritmusunk mindig felhasználja az előzőleg kapott élek eredményeit. Ezután a kapott egyenesdarabokból állitjuk össze a látványgráfot. Ez az eljárás gyors és célratörő, igy alkalmas arra, hogy egy R-10 számitógépen real-time módon /néhány másodperc alatt/ előállitsa a látványgráfot. Ez az egyszerü módszer azonban csak nagyon kevéssé zajos képekre müködik eléggé megbizhatóan, zajosabb képek esetén

biztosabb, több önkontrollal rendelkező algoritmust kell kidolgozni. Ahhoz, hogy egy ilyen algoritmus is real-time módon működhessen, ugy kell azt megtervezni, hogy a végrehajtásán egyidejüleg több processzor dolgozhasson. Ezt a feladatot oldjuk meg a dolgozat 3. fejezetében.

2.1. Élsorozatok összeállitása, csomópontok keresése

Az algoritmus müködésének lényege, hogy mindig specifikál egy ablakot, amiben egyenesdarabot /élet/ keresünk /az él és egyenesdarab szavakat ebben a részben szinonimákként használjuk/. A következő ablakot mindig ugy vesszük fel, hogy az előző él várható folytatása oda essen. Minden megtalált él után körülnézünk, hogy nem jutottunk-e egy korábban talált él közelébe, és igy találjuk meg a csomópontokat.

Az egyes ablakokban az egyenesdarabokat az 1.1. ill. 1.2. pontokban leirt algoritmusokkal keressük. A TV második /csak élpontokat adó/ üzemmódjával kapott kép és a lézerkép feldolgozásának módja ezek után teljesen azonos. A TV első üzemmódjánál kapott képmátrix feldolgozása is csak az algoritmus 1. lépésében különbözik enyhén, a továbbiakban arra is érvényes minden lépés.

Az algoritmusban n x n képpontból álló ablakokban keresünk egyenesdarabokat, gyakorlatilag a lézerképnél n =8 volt, a TV-képnél $5 \leq n \leq 9$ értékkel próbálkozunk, a kép zajosságától függően.

1. lépés: Keressük meg az első olyan ablakot, amelyikben egyenesdarab található.

A TV második üzemmódjából kapott kép és a lézerkép esetén elég csak az inputból kapott pontok köré állitani ablakokat, és azokban vizsgálódni. Ezt ugy tarthatjuk számon, hogy mindig egy pointert állitunk az utolsó már vizsgált input képpontra.

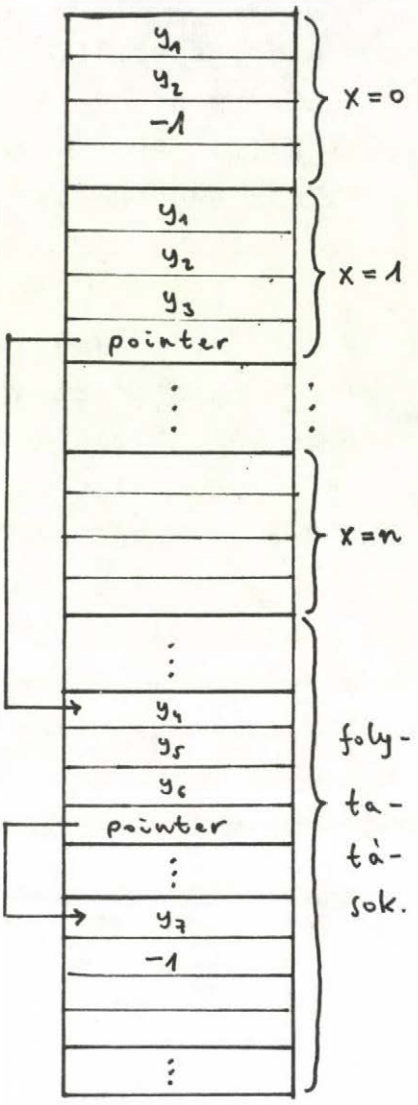
Ebben az esetben minden olyan ablakra, amelyben egyenesdarabot találunk, az inputból kapott képből az ablakban lévő képpontokat kihuzzuk, igy elkerüljük azt, hogy egy korábban már megtalált élet még egyszer vizsgáljunk.

A képmátrixban sajnos nem áll módunkban a "használt" képpontokat kihuzni, hiszen ez uj éleket okozna az ablak határán. Ezért a képmátrixban az 1. lépésnél mielőtt egy ablakban egyenesdarabot keresnénk, meg kell vizsgálni, hogy a vizsgálandó ablak nem metsz-e egy korábban már feldolgozott ablakot. Ez a vizsgálat a 4. lépésben leirt módon történik. A képmátrix esetén tehát ablakokkal szkenneljük végig a képet, igy keresünk kezdő egyenesdarabot. Ezért algoritmusunk gyorsabban müködik a TV második üzemmódjából kapott képekre, mint a képmátrixra. Ha már nem találunk több uj élet, a 6. lépés következik.

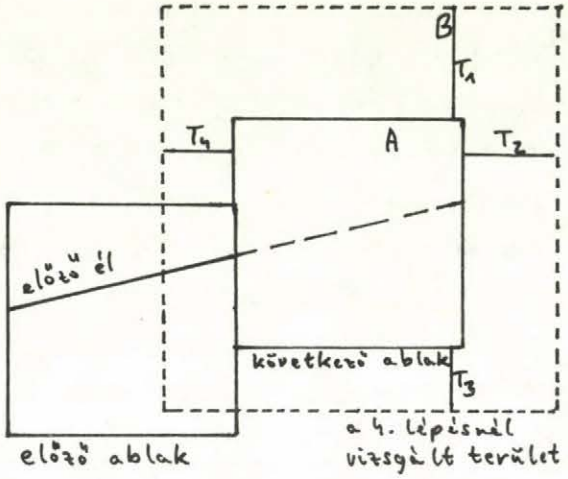
2. lépés: Az utoljára talált egyenesdarabot fellistázzuk a következő módon: minden egyenesdarabnak 5 szó felel meg az egyenesek listájában. Az első két szó annak az ablaknak a középpontjának x-, ill. y-koordinátája, amelyikban az élet találtuk. Ezzel fogjuk helyettesiteni az egyenesdarab végpontjait, ami ugyan enyhe pontatlanság, de nem zavaró, és sok helyet takarit meg. A harmadik szó az él iránytangensét tartalmazza, az 1.1. pontban leirt transzformált alakban. A negyedik szó egy jelzőbit-gyüjtemény, amelyből bizonyosokat már most értelmezünk, másokat később. Az első bit /a_1/ azt jelzi, hogy az él uj élsorozat kezdete-e, vagy folytatása az előző élnek. A következő bit /a_2/ azt mutatja, hogy az él csomópontba vezet-e. /Ennek jelentése a 4. lépés leirásánál válik világossá./ Ha ez a bit igaz, akkor az 5. szó egy pointert tartalmaz arra az egyenesdarabra, amelyiknél a csomót találtuk. A harmadik bit /a_3/ azt jelzi, hogy ez az él egy olyan ablakban van-e, amelyik csomópontot tartalmaz /ld. 5. lépés/. Az a_4 bit azt jelzi, hogy ez az él folytatódik-e, vagy az utolsó eleme egy élsorozatnak. A további biteknek a 6-8. lépésben adunk értelmet.

3. lépés: A gyors keresés érdekében külön, speciális módon is fellistázzuk azoknak a négyzeteknek a középpontját, amelyekben élet találtunk. A lista szervezését a 2.1.1. ábra mutatja. Elősz̈ör minden x-koordináta számára rezerválunk négy helyet, és minden első helyre -1-et irunk végjelként. /-1 abszurd lenne y--koordinátának./ Megjegyezzük, hogy egy ablak középpontjának x- és y-koordinátáját mindig a képfelbontás koordinátarendszerében mérjük, és mindig egész szám - ha az ablak oldala páratlan hosszu, lefelé kerekitünk. A lista végén rezerválunk még valamennyi üres helyet. Az x_i, y_i koordinátáju középpontot ugy jegyezzük fel, hogy az x_i koordináta számára rezervált helyek közül az első üresre irjuk y_i-t, a -1 helyébe. A -1-est a következő helyre toljuk. Ha a -1-es éppen a rezervált helyek negyedikén volt, akkor arra a helyre nem az y_i kerül, hanem egy pointer a lista végén rezervált helyek közül az első üresre, és ugyanott még lefoglalunk 4 ujabb helyett a szóbanforgó x_i koordináta számára. Ezek első helyére kerül most y_i, és a másodikra a -1. Igy tehát az egyenesre rezervált négy hely csak 3 olyan ablak feljegyzésére elegendő, amelynek x-koordinátája x_i, a negyedik annak jelzésére szolgál, hogy kiterjesztettük-e tovább az x_i-hez tartozó listát. Ha 6-nál is több x_i koordinátáju ablak jön le, a listát ugyanigy terjesztjük tovább.

Ezzel a listaszervezéssel elpazarolunk a kép felbontásától függően 1-2,5 k szó helyet, viszont annak ellenőrzése, hogy egy adott (x_o, y_o) pont körül találtunk-e már korábban élet /csomópontkeresés!/ az ablak 2. lépésnél született leirását használva k^2 nagyságrendü keresést igényelne /k éldarab esetén, mivel minden élre kell ilyen keresést végrehajtani/, igy pedig $k^{3/2}$ nagyságrendü keresés elegendő, és a konstans faktor is kisebb. Egyszerübben: 512 x 512-es képfelbontás esetén átlagos bonyolultságu képre az első esetben az egész algoritmus ideje 70-80%-át a következő, 4. lépésnél töltené, mig igy kevesebb, mint 10%-át.



2.1.1. ábra



2.1.2. ábra

4. lépés: Megvizsgáljuk, hogy az utoljára talált éldarab csomópontba fut-e be, azaz hogy egy korábban talált élhez vezet-e. Az utolsó éldarabot ugy irányitjuk, hogy elfele mutasson az utolsó előttitől /tehát az utolsó élnek az utolsó előttihez közelebbi végpontját tekintjük elejének, és a másikat a végének/. Ha az utolsó él egy uj élsorozat első tagja, /tehát az 1. lépés eredményezte/, akkor mindkét lehetséges irányitással próbát teszünk. Legyen A az a négyzet /2.1.2. ábra/, amelynek, ha az utolsó él egyenesen folytatódna, épp a közepén menne át /A szintén n x n-es ablak/, és a 3. lépésnél leirt listát használva vizs-

gáljuk meg, hogy a 2.1.2. ábrán mutatott B négyzeten belül található-e olyan ablak középpontja, amelyben korábban élet találtunk. /Kivéve az utolsó előtti él ablakának középpontját, hogy egészen patalógikus zajok se okozhassanak végtelen ciklust./ A 2.1.2. ábrával értelmezett T_1, T_2, T_3, T_4 paraméterek választása, mint a gyakorlati kisérletek mutatták, igen lényeges, ugyanis ha tul nagyra választjuk őket, akkor egymást valójában nem metsző vonalakra kijelenthetjük, hogy metszik egymást, ha pedig a T_i-k tul kicsik, akkor esetleg létező metszéspontokat nem észlelünk. A legjobb eredményeket

$$T_1 = T_2 = \frac{n}{2} + 2 \qquad T_3 = T_4 = \frac{n}{2} + 1$$

választással kaptunk. /n az ablak oldalhossza./

Ha a B négyzetbe több korábbi ablak középpontja is beleesik, válasszuk ki azt, amelyiknek az utolsó él egyenesétől való távolsága a legkisebb. A B-ben talált ablak középpontját fogjuk kijelölni csomópontnak, igy billentsük be ennél az élnél a 2. lépésben leirt a_3 bitet. Ekkor az utolsó egyenesdarabnál bebillentjük az a_2 bitet, és a listában az eleve üresen hagyott 5. helyet kitöltjük a megfelelő pointerrel. Ha csomópontot találtunk, az utolsó élsorozatot nem is próbáljuk folytatni, /noha esetleg lehetne, ugyanis, ha magasabbfoku csucsba jutottunk, de ekkor a többi idefutó élet később találjuk meg/, hanem uj élsorozatot próbálunk kezdeni: az 1. lépésnél megyünk tovább. Ha nem futottunk csomópontba, az 5. lépés következik.

Az élsorozatok első elemeinél /azaz mindig az 1. lépés után/ ha az él mindkét végén csomóponthoz jutunk, akkor az élet két példányban jegyezzük le és mindkét végén értelemszerüen elvégezzük a szükséges adminisztrációkat. /Tehát, ha az a_1 és a_2 bit is igaz, az azt jelenti, hogy az élsorozat rögtön csomóból indul/. Ez az enyhe megfejelés a későbbiekben semmi zavart nem okoz, és egységes listakezelést tesz lehetővé. Ha élsorozatot kezdő élnek csak az egyik végpontjánál találunk csomót, azt

adminisztráljuk, és a másik irányba folytatjuk az 5. lépéssel. Ha az egyik végén sem találunk csomót, akkor az 5. lépés következik, az él tetszőleges irányitásával.

5. lépés: Megpróbáljuk folytatni az élsorozatot. Először is a 2.1.2. ábrán mutatott A ablakban keresünk uj egyenesdarabot.

Ha az A ablakban nem találtunk élet, átfedő ablakokkal körbejárjuk az utolsó él ablakját, igy keressük meg a folytatást. /Természetesen a képmátrix esetén vigyázva arra, hogy nehogy az utolsó előtti élet találjuk meg ismét./

Ha sikerült folytatást találni, a 2. lépés jön, ha nem, akkor az utolsó él a_4 bitjét bebillentjük, és az 1. lépéssel megyünk tovább.

***

"A csillagocskák is pihentetik az olvasó szemét, értelmét, nem kell mindig az uj számjegy erőteljesebben tagoló hatása."

(Thomas Mann: Doktor Faustus)

***

Az eddigiekben megtaláltuk az input képben az összes élet, és fellistáztuk őket ugy, hogy a csatlakozó egyenesdarabok egymás után vannak.

Definició: Csomópontoknak nevezzük a /leendő/ látványgráf elsőfoku és legalább harmadfoku csucsait. Töréspontoknak fogjuk mondani a másodfoku csucsait. Egy csomópont fokszámának a belőle kiinduló /ill. benne végződő/ élsorozatok számát nevezzük.

A töréspontoknak egyenlőre listáinkban semmi nyoma, ezeket csak a következő pontban találjuk meg. A csomópont-gyanus helyek viszont benne vannak listáinkban: azok az ablakok, amelyeknél az a_1, a_3 vagy az a_4 bit igaz. Ezek azonban még nem a végleges cso-

mópontok: mint látni fogjuk még változhatnak.

6. lépés: Megvizsgáljuk, hogy az egyenlőre elsőfokunak tünő csomópontok /ahol a_1 vagy a_4 igaz, de a_2 és $a_3$nem/ nem vezetnek-e esetleg mégis csomóhoz. Ezekre az élekre tehát megismételjük a 4. lépést, jóval nagyobb T_i-ket véve. /Pl. $T_1 = T_2 = T_3 = T_4 = n+1$ választással./ Ez a lépés alkalmas arra, hogy az input képen látható konturnak a zajokból adódó kisebb folytonossági hibáit kijavitsa.

7. lépés: Azokat az élsorozatokat, amelyek egyetlenegy egyenesdarabból állnak, és legalább az egyik végük elsőfoku, zajnak nyilvánitjuk, és töröljük az élek listájáról. A törlést a bitgyüjtemény ötödik bitje /a_5/ jelzi.

8. lépés: Minden csomópont-gyanus ablakról ellenőrizzük, hogy valóban csomópont-e. Több okból előfordulhat ugyanis, hogy ez nem áll:

a/ a 7. lépésnél kitöröltük azt az élet, ami őt harmadfoku csuccsá tette;

b/ a 6. lépésnél kiderült két végpontról, hogy egymásba futnak /s igy egyiknél az a_2, másiknál az a_3 bit igaz lett/;

c/ a 4. vagy 6. lépésnél két élsorozat-kezdetről kiderült, hogy egyik a másikba fut /és ettől szintén bebillentek a megfelelő a_2 ill. a_3 bitek/;

d/ hasonlóan összekötődött egy kezdet és egy vég. Ezen belül:

d.1/ a kezdetnél billent a_3 és a végnél a_2,

d.2/ forditva, a kezdetnél a_2 billentés és a végnél a_3.

Célunk az, hogy listánkból könnyen és egyértelmüen kiolvashatók legyenek a csomópontok és az őket összekötő élsorozatok; ennek megvalósitása még némi gondosságot igényel.

Az a_6 bit jelezze azt, hogy azokon a helyeken, ahol a_3 áll, valójában másodfoku csomópont van-e. Ennek ellenőrzéséhez annyiszor, ahányszor a_3 igaz, végig kell szaladni az élek listáján, és vizsgálni az 5. helyen álló pointereket. Szerencsére egy szóbajöhető ábrán nincs nagyon sok csomópont, igy ez nem vesz el számottevő időt.

Ahol a_6 igaz, és sem a_1, sem pedig a_4, ott a felsorolt esetek közül csak a/ állhat, igy a_3 és a_6 visszabillentése után a lista nyomban korrekté válik.

Ha valahol a_1, a_2 és a_6 is igaz /és persze igy a_3 is/, akkor ott ezt az élet egyszerüen kitöröljük, és ugy irjuk át a listát, mintha az ebbe belefutó él rögötn oda futott volna, ahová ez futott bele. /Ez az eset csak igen nagy zaj következtében jöhet létre, igy rögtön azt is kiszürjük./

Most már azokon a helyeken, ahol a_6 igaz, az 5. helyen biztosan nincs pointer. /Ezeken a helyeken persze vagy a_1, vagy a_4 igaz./ Az 5. helyre állitsunk be ezeknél az éleknél olyan pointereket, amelyek a b/-d/ eseteknek megfelelően az élsorozat folytatására mutatnak. Az a_7 bit jelezze azt, hogy ha a pointer szerint akarjuk folytatni az élsorozatot, akkor onnan előrefelé, vagy hátrafelé kell-e olvasni egymásután az élek listáját, hogy az élsorozat folytatását sorban megkapjuk. Végül ezeken a helyeken is állitsuk vissza 0-ra a_3-at, hogy a_3 csak a tényleges csomók helyét jelezze.

9. lépés: Most már tetszőleges élsorozat egyenesdarabjai egymás után kiolvashatók a listából, csomóponttól csomópontig. Utolsó problémánk az, hogy ha valahol az élek egyetlen zárt hurkot alkotnak, akkor /mivel épp az előbb sikerült kiirtani belőle az álcsomópontot/ csomópont hiján nem tudjuk honnan elkezdeni az élsorozat olvasását. Ezért az ilyen elágazásmentes zárt hurkokba mégis belerakunk egy álcsomópontot, ezt az esetet jelezze bitgyüjteményünk utolsó tagja, a_8.

Az igy kapott listastrukturából az ábra csomópontjai és az őket összekötő élsorozatok kiolvashatók.

2.2. A konturvonalak meghatározása

A konturvonalakat /azaz a látványgráf éleit/ egyenes szakaszokból és ivekből állitjuk össze. Ivnek nevezünk a továbbiak-

ban egy olyan görbe vonalat, amely vagy végig konvex, vagy végig konkáv. /Nincs "inflexiós pontja"/. Egy ivet három pontjával: kezdőpontjával, végpontjával és egy tetszőleges közbülső pontjával irunk le, és a későbbi feldolgozásra bizzuk annak a meghatározását, hogy az iv a tárgy milyen köriv-élének a képe. Két csomópont között az előző pontban megkapott egyenesdarabok sorozatát jelölje $\{s_k;\ k=1,2,\ \dots\ ,\ n\}$, ezek az állistánkból sorban kiolvashatók. /n tehát a két csomópont közötti élsorozatban az élek száma./ Legyen α_k az s_k egyenesdarab irányszöge az 1.1. pontban leirt formában, és $\Delta_k = \alpha_{k+1} - \alpha_k$. A konturvonalak összeállitását három lépésben végezzük.

a/ Legyen T egy előre meghatározott küszöbérték. /2 és 4 között változtatgattuk, pl. T=3 megfelel 7 x 7 -es ablakméret esetén./ Legyen K_o az élsorozat kezdő csomója, $K_o = O$ és

$$k_i = \max\left\{ k;\ \sum_{j=k_{i-1}}^{m} \Delta_j < T\ ,\ \text{minden}\ m < k \leq n - \varkappa \right\}$$

Igy az utolsó $k_\ell = n$. Legyen továbbá K_i az s_i él ablakának középpontja, /ezzel a ponttal reprezentáltuk az élet/, és K_ℓ az élsorozat végén lévő csomó.

b/ Legyen $l_o = O$, és sorra minden r-re

$$l_{2r+1} = \min\left\{ l;\ l_{2r} \leq l \leq n-3\ ,\ \operatorname{sign}\Delta_l = \operatorname{sign}\Delta_{l+1} = \operatorname{sign}\Delta_{l+2} \neq 0 \right\}$$

$$l_{2r+2} = \max\left\{ l;\ l_{2r+1} \leq l \leq n\ ,\ \operatorname{sign}\Delta_l = \operatorname{sign}\Delta_{l_{2r+1}} \right\}$$

Előfordulhat, hogy ilyen $r \neq O$ nem is létezik. Legyen L_p az s_{l_p} él reprezentáns pontja, ha $l_p \neq O$ és $l_p \neq n$. Ha $l_p = O$, L_p legyen az élsorozat kezdő csomója, ha pedig $l_p = n$, akkor L_p az élsorozat végén lévő csomó.

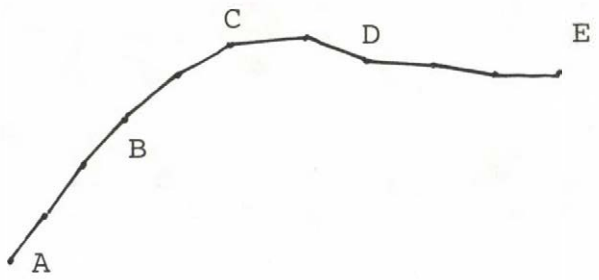
c/ Töréspontnak /ld. 2.1. 5. lépés előtt/ vesszük az összes L_i pontokat és az összes olyan K_j pontokat, melyekre

$$k_j \notin \bigcup_r \bigcup_{i=l_{2r+1}}^{l_{2r+2}} i$$

Az L_{2r+1} és L_{2r+2} pontok között iveket észlelünk, az iv közbülső pontjának egy $s_{1_{2r+1}}$ és $s_{1_{2r+2}}$ közötti él reprezentáló pontját tekintjük. A többi egymásutáni töréspont között egyenesszakaszokat veszünk fel, immár a kész látványgráf első változatában. Egy ivet konvexnek, ill. konkávnak mondunk, aszerint, hogy a hozzá tartozó egyenesszakaszoknál $\Delta_r > 0$ vagy $\Delta_r < 0$.

Az algoritmus értelme röviden a következő: egyenesszakaszokat állitunk össze az élekből mindaddig, amig iránytangenseik különbsége egy küszöbérték alatt van, és azért vesszük végig a különbségek összegét, hogy egy-egy hibás /zajos/ él korrigálódhasson. /Ezzel persze nagyon tompa szögü egyeneseket egybeolvaszthatunk./ Ivet észlelünk akkor, ha legalább négy egymásutáni egyenesszakasz ugyanarrafelé hajlik, azaz, ha a konturvonal második deriváltjának előjele huzamosabban azonos.

Bizonyos esetekben még emberi megfigyelő is nehezen dönti el egy élsorozatról, hogy az törött vonal-e, vagy köriv /zajos, tehát kicsit hibás élek esetén/. Egy ilyen esetet mutat be pl. a 2.2.1. ábra. Nehéz eldönteni, hogy mi a helyzet: AC és CE is



2.2.1. ábra

egyenes, vagy AB és DE egyenes, és BD köriv. Ez a kétség az algoritmusba is beépithető, ha a signum-függvényt a következő módon definiáljuk:

$$\operatorname{sign} x = \begin{cases} 0 & \text{ha} \quad -\varepsilon \leq x \leq \varepsilon \\ 1 & \text{ha} \quad x > \varepsilon \\ -1 & \text{ha} \quad x < -\varepsilon \end{cases}$$

ahol ε lehet pl. 1/2 vagy 1.

Ez az algoritmus azonban még igy is tulságosan determinisztikus, sok esetben jobb a későbbi feldolgozásra hagyni annak eldöntését, hogy pl. egy bizonytalannak értelmezhető élsorozat valójában kör-e, vagy egyenes, vagy hogy egy egyenes-e, vagy kettő. A felismerés során ugyanis később, a környező vonalak ismeretében ezek a kérdések sokkal könnyebben eldönthetők. A 3.4. pontban egy olyan algoritmust ismertetünk, amely az egyes élsorozatok vonalakként való értelmezésére bizonytalan esetben alternativákat is képes szolgáltatni.

3. Párhuzamos és kvázi-párhuzamos képfeldolgozó algoritmusok

3.0. Bevezetés

Párhuzamos algoritmusoknak olyan eljárásokat nevezünk, ahol bizonyos /önmagukban is számitógépszerüen müködő/ elemek, amelyek össze vannak kötve más elemekkel /szomszédaikkal/, az uj értéküket a szomszédaik aktuális értékeinek függvényében, egyidejüleg veszik fel. Ezzel szemben, soros algoritmusnak olyan eljárásokat nevezünk, ahol az elemek egymás után veszik fel szomszédaiktól függő uj értékeiket, ahol tehát a szomszédok közül némelyek már átestek a feldolgozás szóbanforgó lépésén, mások pedig még nem. Soros algoritmusok implementálására természetes eszközök jelenlegi számitógépeink, mig a párhuzamos algoritmusok hatékony implementálása merőben uj architekturáju számitógépeket igényel. Ilyen ujfajta számitógépek épitésének lehetőségét először Neumann János vetette fel hires sejtautomata tanulmányában [52].

Hatékony párhuzamos algoritmusok tervezése általában igen nehéz feladat, mert az egész megoldandó feladatot egyszerre kell kezelni, ugyanis, /mivel minden elem egyszerre változik/, nehezen definiálhatók egyszerübb részfeladatok. Másrészt párhuzamos algoritmus implementálására alkalmas hardware épitésének nehézsége az, hogy nagyon sok, önmagában is számitógépszerüen müködő elemet kell összeépiteni. /Pl. már a 3.2.1. vagy 3.2.2-ben leirt algoritmusokhoz 256 x 256-os képmátrixra is kb. 65000 darabot./ Ilyen kifejezetten képfeldolgozásra épitett kisérleti hardwareket irnak le Kruse [41], Shi-Kuo Chang [71], továbbá Ramesh és Fu [59].

Definiálunk egy olyan algoritmus-tipust, amely alkalmas kompromisszum a soros és párhuzamos eljárások között. Kvázi-párhuzamos algoritmusnak nevezzük az olyan algoritmusokat, amelyek lehetővé teszik, hogy bizonyos müveleteket egyidejüleg,

egymástól függetlenül /ebben különbözik az igazi párhuzamos algoritmustól/ hajtsunk végre különböző elemeken. Ez azt jelenti, hogy n processzor alkalmazása az egyidejüleg végrehajtott müveletek együttes idejét n-edrészére csökkentheti, de nem szükséges annyi processzor, ahány elemre a müveletet el kell végezni. n = 1 processzor esetén a kvázi-párhuzamos algoritmusok sorossá fajulnak. A kvázi-párhuzamos módszerek nem tartalmazzák azokat az izgalmas lehetőségeket, ahogyan a párhuzamos gépekkel az agy müködését próbálják szimulálni, viszont ilyen hardware sokkal egyszerübben és olcsóbban megvalósitható.

A képfeldolgozás területén különösen sok lehetőség adódik párhuzamos és kvázi-párhuzamos feldolgozása. Az 1.1. és 1.2. pontban leirt élkereső eljárás például egyidejüleg elvégezhető a kép különböző részeire, a 2.1. pontban leirt algoritmus azonban szigoruan kihasználja, hogy az éldarabokat sorosan, egymás után kapjuk meg. Azt az algoritmust ugyanis egy soros müködésü számitógépre terveztük, és éppen az volt a cél, hogy ezen a gépen müködjön gyorsan. A párhuzamos feldolgozás jövőbeli szükségességét mutatja az is, hogy a 2.1. pontban leirt algoritmus ideje 90%-át az éldarabok detektálásával tölti, ami kvázi-párhuzamosan is végezhető. /Az emberi agy retinasejtjei is kvázi--párhuzamosan müködnek./ A soros feldolgozásnál viszont az algoritmus tervezését megkönnyitette azt, hogy a következő élet az előzők ismeretében kereshettük, amit az élek párhuzamos keresése esetén nem tehetünk meg. Ebben az értelemben tehát a párhuzamos eljárás "butább", mint a soros, viszont a tetemes időnyereség egy részét felhasználhatjuk ennek ellensulyozására, sőt ezzel a párhuzamos algoritmus hatékonysága sokkal nagyobb mértékben növelhető. Nem kell ugyanis elfogadni mindig az első adódó éldarabot, hanem az összes élek ismeretében kiválaszthatók /alkalmasint továbbra is kvázi-párhuzamos algoritmusokkal/ a konturvonalakat optimálisan közelitő élsorozatok.

Kvázi-párhuzamos képfeldolgozó eljárásra Stefanelli és Rosenfeld [77] vékonyitó algoritmusa lehet példa. Itt a feladat az, hogy vonalszerü bináris képeket /pl. nyomtatott karakterek vagy kromoszómák képeit/ vékonyitsunk le ugy, hogy eredményül

egyetlen képpont szélességü vonalat kapjunk. Stefanelli és Rosenfeld [77], ill. Rosenfeld és Davis [64] olyan operációkat definiálnak egy pont 3 x 3 négyzetnyi környezeteiben, amelyek bizonyos esetekben eltüntetik a képpontot, és amelyek eredményeként /mint azt bebizonyitják/ az eredeti kép valóban 1 pont szélességü vonalakra vékonyodik le. Bizonyitásukat megvizsgálva kiderül, hogy algoritmusuk kvázi-párhuzamos módon is müködik, operációikat bármelyik pont környezetén, tetszőleges sorrendben, akár részben vagy teljesen egyidejüleg végrehajtva az ábra levékonyodik, és a vonalak összefüggése nem szakad meg.

Négyzetrácson digitalizált bináris képen egy S halmaz összefüggőségére két különböző definició is adható: egy S halmazt 4-összefüggőnek nevezünk, ha bármely két pontja között megadható S-beli elemeknek egy olyan sorozata, amelynek első, ill. utolsó tagja a két szóbanforgó pont, és bármely két egymásutáni (i,j) és (h,k) pontra

$$d_4\left[(i,j),\ (h,k)\right] = |i-h| + |j-k| \leq 1$$

S-et 8-összefüggőnek nevezzük, ha fenti definicióban a d_4 távolságot a következő d_8 távolsággal helyettesitjük:

$$d_8\left[(i,j),\ (h,k)\right] = \max\{|i-h|,\ |j-k|\}$$

Hozzá kell azonban tenni, hogy ha az 1-esek között a 4-összefüggőséget tekintjük definiciónak, akkor a 0-sok között a 8--összefüggőséget kell tekinteni és forditva. Erre azért van szükség, mert ellenkező esetben kellemetlen ellentmondásokra juthatunk: nem lesz igaz pl. az Euler-tétel, sem a Jordan-tétel. Az adott definiciók mellett viszont mindkét fontos tételnek megvan a digitális megfelelője, ha a zárt görbe fogalmát értelemszerüen difiniáljuk. Diszkretizált terek topológiai tulajdonságait Mylopoulos és Pavlidis [50] vizsgálták, sikerült egy konzisztens dimenziófogalmat is definiálniuk ilyen terekre.

Valódi párhuzamos képfeldolgozó algoritmusra /aholis minden operációt egyszerre minden képponton el kell végezni/ példa Levi-

aldi [42] összehuzó algoritmusa. Itt a feladat az, hogy egy bináris képben, ahol 1-esek ábrázolják a tárgy, 0-k a háttér pontjait, számoljuk meg az összefüggő 1-esekből álló komponenseseket, azaz másképp fogalmazva, minden összefüggő tárgyat reprezentáljunk egy ponttal /huzzuk össze ponttá/.

Egy összehuzó algoritmust például ugy tervezhetünk, hogy sorra töröljük azokat az 1-eseket, amelyek törlése következtében az illető pont környezetében az 1-esek összefüggősége nem változik. Izzo és Coles [39] megmutatják, hogy elég minden 1-es törlésénél a pont 3 x 3-as környezetét vizsgálni, és leirnak olyan logikai szabályokat, hogy ha azok szerint a szabályok szerint sorra töröljük a törölhető 1-eseket, akkor végül minden összefüggő komponens egyetlenegy izolált 1-esre huzódik össze. Algoritmusukat kidolgozták a 4- és 8-összefüggő esetre is. A továbbiakban mindig a 4-összefüggő esetet vizsgáljuk, azzal a megjegyzéssel, hogy a komplementerre áttérve minden átvihető a 8-összefüggő esetre is.

Párhuzamos összehuzó algoritmus tervezése már nagyobb körültekintést igényel, hiszen egy párhuzamosan elkalmazott összehuzó operáció könnyen eltüntethet egyszerre minden 1-est az ábrából. Rosenfeld [63] bebizonyitotta, hogy egy párhuzamos összehuzó algoritmusnak minden pontnak legalább 5 x 5-ös környezetét figyelembe kell vennie, feltéve, hogy az algoritmus csak bizonyos képpontokat töröl, de uj 1-eseket nem generál. Nagyobb környezet vizsgálata kell például azért, hogy nehogy két, csupa 0-kal körülvett, szomszédos 1-es egyszerre kitörölődjön. Rosenfeld 5 x 5-ös ablakokat használva konstruált is egy párhuzamos összehuzó algoritmust.

Levialdi [42] észrevette, hogy ha azt is megengedjük, hogy uj képpontok is generálódjanak az algoritmus során, akkor már 2 x 2-es környezetet vizsgálva is megadható /méghozzá igen egyszerü/ összehuzó algoritmus. Levialdi algoritmusa a következő: Legyen az (i,j) képpontban a kép értéke az n-edik lépés után $a^n_{i,j}$ és minden (i,j)-re egyszerre legyen

$$a_{i,j}^{n+1} = \Phi\left(a_{i,j}^{n}, a_{i+1,j}^{n}, a_{i,j+1}^{n}, a_{i+1,j+1}^{n}\right) =$$

$$= h\left[h\left(a_{i,j+1}^{n} + a_{i,j}^{n} + a_{i+1,j}^{n} - 1\right) + h\left(a_{i,j}^{n} + a_{i+1,j+1}^{n} - 1\right)\right]$$

ahol

$$h(x) = \begin{cases} 0, & \text{ha} \quad x \leq 0 \\ 1, & \text{ha} \quad x > 0 \end{cases}$$

Ekkor a Φ függvény párhuzamos alkalmazásáról bebizonyitható, hogy korrekt párhuzamos összehuzó algoritmus.

Rao, Prasada és Sarma [60] olyan, Levialdiéhoz hasonló algoritmust irt le, amely ugyan 3 x 3-as környezeten dolgozik, de az összefüggő komponensekből kapott izolált 1-esek a további lépésekben sem tünnek el, és minden komponenst a széle felől a közepe felé huz össze. /Levialdi algoritmusa minden komponenst a bal alsó sarkára huz össze./ Levialdi algoritmusa valódi párhuzamos eljárás, ugyanis egyszerre minden elem /képpont/ értéke változhat minden lépésben.

Ha az eljárást sorosan, vagy kvázi-párhuzamosan akarnánk végezni, azaz nem egyszerre minden elemre, akkor például a 3.0.1. ábrán látható képből a középső elemet kitörölhetnénk, tehát a komponens összefüggősége megszakadna. Φ párhuzamos alkalmazása a 3.0.1. ábrára a 3.0.2. ábrát eredményezi.

1	0	0
0	1	0
0	0	1

3.0.1. ábra

0	0	0
1	0	0
0	1	0

3.0.2. ábra

Rosenfeld 5 x 5-ös ablakokon müködő algoritmusa [63] viszont, noha párhuzamosan is végezhető, sorosan, vagy kvázi-párhuzamosan alkalmazva is korrekt összehuzást ad, /bár az összefüggő komponensek esetleg más pontra huzódnak össze/, igy ez az

eljárás valójában kvázi-párhuzamos.

Igazi párhuzamos algoritmusokra egy másik példa a relaxációs operációk. A relaxációs operációk alapgondolata a következő:

Legyen $A = \{a_1, \ldots, a_n\}$ objektumok egy halmaza, $\Lambda_i = \{\lambda^i_1, \ldots, \lambda^i_{m_i}\}$ pedig minden i-re az a_i objektumhoz rendelhető cimkék halmaza. /Az a_i objektum lehetséges szemantikus értelmezései./

Minden λ^i_j cimkéhez legyen hozzárendelve egy $0 < p^{(0)}_{i,j}(\lambda^i_j) < 1$ valószinüség, ugy, hogy

$$\sum_{j=1}^{m_i} p^{(0)}_{i,j}(\lambda^i_j) = 1 \qquad i = 1, 2, \ldots, n$$

Legyen minden i,j párra definiálva egy $r_{ij}(\lambda,\lambda')$ kompatibilitási függvény ugy, hogy

$$r_{ij}(\lambda, \lambda') : \Lambda_i \times \Lambda_j \rightarrow [-1, 1] \; ; \; \lambda \in \Lambda_i, \; \lambda' \in \Lambda_j$$

Ezek az r_{ij} számok azt fejezik ki, hogy a különböző cimkék /azaz értelmezések/ valószinüsitik, vagy éppen kevésbé valószinüvé teszik egymást.

A relaxációs operáció egy F operátor, amely mindig az előző $p^{(k)}_{i,j}(\lambda^i_j)$ valószinüségek és az r_{ij} kompatibilitási függvények alapján meghatározza a következő $p^{(k+1)}_{i,j}(\lambda^i_j)$ valószinüségeket. Az eredmény a határértékként kapott $p^{(\infty)}_{i,j}(\lambda^i_j)$ valószinüségek. Ezek általában jobb lehetőséget adnak az eredeti cimkék értelmezésére, mint a kiinduló /esetleg ellentmondásos/ $p^{(0)}_{i,j}(\lambda^i_j)$ valószinüségek.

A relaxációs operációk hasonló gondolatot valósitanak meg, mint a sztochasztikus approximáció közismert esetei, de mig az utóbbiak a függvényértékeket változtatják lépésről-lépésre,

a relaxációs operációk a különböző értelmezések valószinüségének értékeit változtatgatják.

A relaxációs operációk gondolatát először Waltz [80] vetette fel, és Rosenfeld, Hummel és Zucker [65] dolgozatukban adták meg korrekt matematikai leirásukat. Ugyanők [84] dolgozatukban több javaslatot tettek a relaxációs operációk felhasználására a képfeldolgozásban.

Például Schachter, Lev, Zucker és Rosenfeld [69] az egyenesdarabokból a hosszabb egyenesek összeállitására javasolják a relaxációs operációk használatát. Itt az a_i objektumok az éldarabok, minden Λ_i a lehetséges irányszögek halmaza, a $p_{i,j}^{(0)}$ valószinüségeket az élek "jóságának" mérőszámából kapják, a kompatibilitási függvényeket pedig minden éldarabhoz csak a szomszédaira és azok szomszédaira értelmezik ugy, hogy a közeli irányszögü éldarabok erősitsék a hasonló irányszög-cimkék valószinüségeit, a nagyon eltérő éldarabok pedig gyengitsék egymás valószinüségeit. A határértékként kapott $p_{i,j}^{(\infty)}(\lambda_j^i)$ értékek körül minden i-re a maximálisat választják az a_i egyenesdarab irányszögének. Ezzel a módszerrel, mint kisérleteik mutatják, szépen "ki tudják simitani" az egyenesdarabokat.

A relaxációs operációk hasonló felhasználásához kapcsolódik Herman, Lentz és Lutz [33] és Haralick [32] munkája is. A relaxációs operációk konvergációját Zucker, Krishnamurthy és Haar [85] bizonyitotta.

A relaxációs operációk tipikusan párhuzamos algoritmusok, hiszen minden elem uj értékét az összes elem régi értéke alapján kell kiszámolni. Ha valahol az uj értékkel számolnánk, előfordulhatna, hogy két cimkéről, amelyek semlegesitik egymást azt kapnánk, hogy az egyik semlegesedik, mig a másik /emiatt/ nem.

Számos párhuzamos képfeldolgozó algoritmus hatékonyságát vizsgálja Cordella, Duff és Levialdi [22].

A 3. rész további pontjaiban egy kvázi-párhuzamos algoritmust dolgozunk ki a látványgráf előállitására. A 3.1. pontban egy kvázi-párhuzamos csomópontkereső algoritmust irunk le. Hasonló feladatot old meg Perkins és Binford [55], de ők adott tipusu csomó-

pontokat kerestek a képben. Freeman és Davis [25] elágazás nélküli konturok töréspontjainak meghatározására adnak meg egy algoritmust.

A 3.3. pontban optimális élsorozatok előállitására dolgozunk ki egy kvázi-párhuzamos eljárást. Ramer [58] dolgozatában szintén optimális élsorozatok keresésével foglalkozik, de algoritmusa szigoruan soros, és célja nem látványgráf előállitása, hanem a képben minél hosszabb összefüggő optimális élsorozat keresése.

3.1. Csomópontkeresés

Csomópontokon a látványgráf legalább harmadfoku, vagy legfeljebb elsőfoku csucsait értjük. A csomópontok megkeresésére egy soros eljárást leirtunk a 2.1. pontban. Most egy kvázi-párhuzamos algoritmust mutatunk be legalább harmadfoku csomópontok keresésére.

Fedjük le az n x m-es képmátrixot k x k méretü ablakokkal ugy, hogy két szomszédos /egymás melletti vagy alatti/ ablak ql pontnyira fedje át egymást. Ilyen módon az ablakok egy s x t elemü rendszert alkotnak, ahol

$$s = \left[\frac{n - q - 1}{k - q}\right] \qquad \text{és} \qquad t = \left[\frac{m - q - 1}{k - q}\right]$$

Minden ablakra egymástól függetlenül /tehát kvázi-párhuzamos módon/ végezzük el az 1.4. ill. 1.5. pontban leirt élkereső algoritmust. Definiáljuk az élmátrixot és a sulymátrixot a következőképpen: mindkét mátrix s x t elemü, és az i-edik sor j-edik eleme az élmátrixban az ablakok i-edik sorának j-edik elemében talált egyenesdarab irányszöge, ha abban az ablakban találtunk egyáltalán élet, és egy speciális jel, ha nem. A sulymátrix megfelelő eleme legyen az itt talált él "jóságára" kapott W mérőszám. /W-t az 1.1. pont végén definiáltuk./

Csomópontkereső algoritmusunk alapgondolata a következő: Vegyünk fel az (x_o,y_o) pont egy U_o környezetében egy (x_o,y_o)-on áthaladó e egyenest és jelöljük $S_e^1(x_o,y_o)$-lal azon egyenesdarabok irányszögének szórását, amelyeket az e egyenes egyik oldalára eső U_o-beli ablakokból nyerünk, $S_e^2(x_o,y_o)$ pedig legyen ez e egyenes másik oldalára eső U_o-beli élek irányszögének szórása.

Legyen

$$S_e(x_o,y_o) = \max \{ S_e^1(x_o,y_o) , S_e^2(x_o,y_o)\}$$

Tegyük fel, hogy az (x_o,y_o) pontban legalább három különböző szürkeségi szintü homogén terület találkozik a képben, /azaz az elkészitendő látványgráfnak itt legalább harmadfoku csucsa lesz./ Ekkor $S_e(x_o,y_o) > 0$, sőt az

$$S(x_o,y_o) = \min_e \{ S_e (x_o,y_o)\}$$

mennyiségre, ahol a minimumot az összes olyan e egyenesre tekintjük, amelyik áthalad (x_o,y_o)-on, és $S (x_o,y_o) > 0$. /Itt használjuk ki, hogy (x_o,y_o)-ban legalább három homogén régió találkozik./ Ha minden (x,y) pontban meghatározzuk az S(x,y) függvényt, akkor az S(x,y) függvénynek a csomópontokban lokális maximumhelyei vannak. Erre az észrevételre alapul csomópontkereső algoritmusunk.

Mielőtt az algoritmus tényleges végrehajtását leirnánk, vegyük észre, hogy a gondolatmenetben az irányszögek szórásának fogalma nem világos, mivel az irányszögértékek közötti különbséget értelemszerüen modulo Π kell érteni, azaz, mivel az 1.4-beli értelmezés szerint irányszög-értékeink -k és k közé esnek, modulo 2k. Ezért ujra definiáljuk a szórás fogalmát, hogy erre az esetre is értelmes legyen. Az egyszerüség kedvéért essenek az irányszög-értékek 0 és 2k közé.

Az S_1, S_2, ... S_r számok szórásának tekintsük azt a σ számot, amelyre

$$\sigma^2 = \min_s \left\{ \frac{\sum_{i=1}^{r} (s - s_i)^2}{r-1} \right\} \qquad (1)$$

ahol $\dot{-}$ a modulo 2k vett különbséget jelöli. Ismeretes az alábbi egyszerü állitás:

1. állitás: Ha $s_1, \ldots, s_r$ valós számok a számegyenesen, akkor

$$\sigma^2 = \frac{\sum_{i=1}^{r} (s' - s_i)^2}{r-1} = \min_s \left\{ \frac{\sum_{i=1}^{r} (s - s_i)^2}{r-1} \right\}$$

ahol $s' = \frac{s_1 + \cdots + s_r}{r}$.

A továbbiakban megmutatjuk, hogy az (1)-ben szereplő minimum kiszámitása r+1 féle szórásérték minimumának meghatározásával megoldható /ld. 3.1.1. ábra/. Legyen minden $1 \leq i \leq r$ -re

$$t_i = \begin{cases} s_i + k , & \text{ha} \quad 0 \leq s_i < k \\ s_i - k , & \text{ha} \quad k \leq s_i < 2k \end{cases}$$

Legyenek továbbá az a_i indexek olyanok, hogy $t_{a_0} = 0 \leq t_{a_1} \leq \ldots \leq t_{a_r} \leq 2k = t_{a_{r+1}}$ fennálljon. /Itt az irányszögekből kapott t_i értékeket még két számmal kiegészitettük, hogy az indexelés egységes lehessen./

2. állitás: Ha $0 < t_{a_i} \leq k$, akkor a $\left[t_{a_{i-1}}, t_{a_i}\right]$ intervallumon

$$\min_{t_{a_{i-1}} \leq s \leq t_{a_i}} \sigma^2(s) \geq \frac{\sum_{i=1}^{r} (s_i' - s_0)^2}{r-1} = \sigma_{a_i}$$

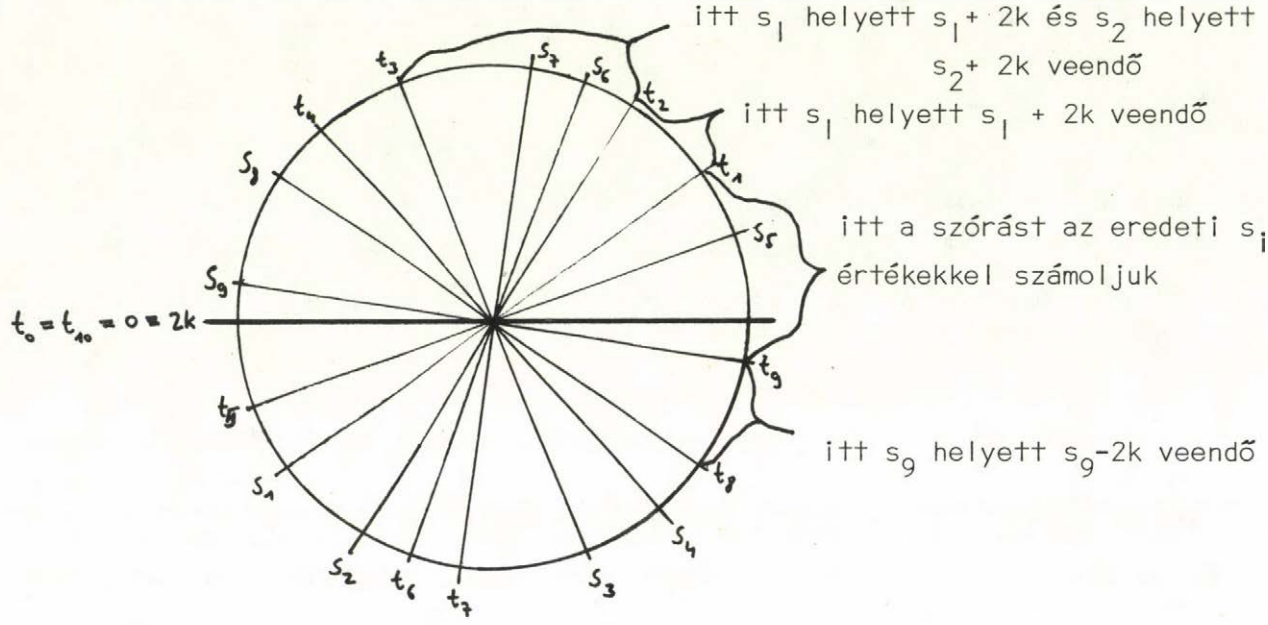
ahol

$$\sigma_s^2 = \frac{\sum_{i=1}^{r} (s \dot{-} s_i)^2}{r-1}$$

$$s_i' = \begin{cases} s_i , & \text{ha} \quad s_i \leq s_{a_i} \\ s_i - 2k, & \text{ha} \quad s_i > s_{a_i} \end{cases} \qquad \text{és} \qquad s_0 = \frac{\sum_{i=1}^{r} s_i'}{r}$$

Egyenlőség akkor és csak akkor áll fenn, ha $t_{a_{i-1}} \leqslant s_o \leqslant t_{a_i}$

Bizonyitás: Egy, a tekintett intervallumba eső s-től mindegyik s_i-nek a $\div$ szerint vett távolsága éppen s'-s, ezért az állitás az 1. állitásból következik. ■



3.1.1. ábra

3. állitás: Ha $t_{a_{i-1}} \leqslant k < t_{a_i}$, akkor

$$\min_{t_{a_{i-1}} \leqslant s \leqslant t_{a_i}} \sigma^2(s) = \frac{\sum_{i=1}^{r} (s_i - s_o)^2}{r-1} = \sigma^2_{a_i}$$

ahol

$$s_o = \frac{\sum_{i=1}^{r} s_i}{r}$$

4. állitás: Ha $2k > t_{a_{i-1}} > k$, akkor a $\left[t_{a_{i-1}}, t_{a_i}\right]$ intervallumon

$$\min_{t_{a_{i-1}} \leqslant s \leqslant t_{a_i}} \sigma^2(s) \geqslant \frac{\sum_{i=1}^{r} (s_i' - s_o)^2}{r-1} = \sigma^2_{a_i}$$

ahol

$$s_i' = \begin{cases} s_i , & \text{ha} \quad s_i \geq s_{a_i} \\ s_i + 2k, & \text{ha} \quad s_i < s_{a_i} \end{cases}$$

és

$$s_0 = \frac{\sum_{i=1}^{\tau} s_i'}{\tau}$$

Bizonyitás: A 3. és 4. állitás ugyanugy látható be, mint a 2. állitás. Egyenlőség itt is akkor és csak akkor áll fenn, ha az aktuális s'-értékek szerinti átlag beleesik a szóbanforgó intervallumba. ■

5. állitás: Ha a $\sigma(s)$ függvény valamelyik $\left[t_{a_{i-1}}, t_{a_i}\right]$ intervallumon felveszi minimumát, akkor ott a 2., 3., ill. 4. állitás közül annál, amelyik erre az intervallumra vonatkozik

$$\min_{t_{a_{i-1}} \leq s \leq t_{a_i}} \sigma^2(s) = \sigma^2(a_i)$$

áll.

Bizonyitás: Ha a $\left[t_{a_{i-1}}, t_{a_i}\right]$ intervallumnak megfelelő s_0 érték valamelyik másik $\left[t_{a_{j-1}}, t_{a_j}\right]$ intervallum belsejébe esne, akkor a $\left[t_{a_{j-1}}, t_{a_j}\right]$ intervallumhoz tartozó s_i' értékkel számitott $\sigma^2_{a_j}$-re $\sigma^2_{a_j} < \sigma^2_{a_i}$ állna, ami $\min_{t_{a_{i-1}} \leq s \leq t_{a_i}} \sigma^2(s) \geq \sigma^2_{a_i}$

miatt ellentmond annak, hogy $\sigma^2(s)$ a $\left[t_{a_{i-1}}, t_{a_i}\right]$ intervallumon veszi fel a minimumát. Ha viszont $t_{a_{i-1}} \leq s_0 \leq t_{a_i}$, akkor a 2-4. állitásokból következik az 5. állitás. ■

A 2-5. állitásokból következik, hogy

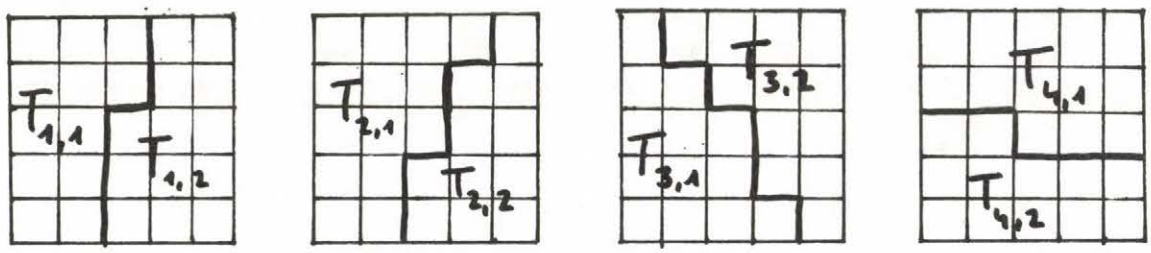
$$\min_{s} \sigma^2(s) = \min_{0 \le i \le \pi} \sigma^2(a_i)$$

tehát az irányszög-értékek szórása $O(r^2)$ müvelettel kiszámitható, és az állitásokból a kiszámitás algoritmusa is kiolvasható.

A csomópontkereső algoritmust a számitógéppel ugy hajtjuk végre, hogy az s x t elemü élmátrix minden (x,y) pontjára kiszámitjuk s(x,y)-t a következőképpen: Az (x,y) pont egy 5 x 5 pontos környezetében a 3.1.2. ábrán ábrázolt $T_{u,v}$ területeken /$1 \leq u \leq 4$, $1 \leq v \leq 2$/ található irányszögeknek kiszámitjuk a $\sigma^2_{u,v}$ szórását, és s(x,y)-t az

$$S'(x,y) = \min_{u} \{ \max_{v} \{\sigma^2_{u,v}\}\}$$

mennyiséggel közelitjük.



3.1.2. ábra

Ezután megkeressük az S'(x,y) lokális maximumhelyeit az (x,y) pontok 3 x 3 -as környezetein ugy, hogy egyenlő értékek közül a baloldalit, ill. a felsőt tekintjük nagyobbnak. Végül csomópontokat detektálunk az eredeti képen a lokális maximumhelyekhez tartozó ablakok középpontjaiban.

Az algoritmus egymástól legfeljebb kb. 10-12 képpont távolságra lévő csomópontokat gyakran egybemos, de ennél távolabbi csomópontokat szépen megtalál. A kisérleti eredményeket a 4. fejezetben mutatjuk be, az ottani ábrákon láthatóak a csomókereső algoritmus eredményei is.

Az eljárás folyamán az éldarabok detektálása, az S'(x,y) értékek kiszámolása és a lokális maximumhelyek meghatározása is az ábra különböző részein egyidejüleg, egymástól függetlenül végezhető, ezért ez a csomópontkereső algoritmus kvázi-párhuzamos.

3.2. Matematikai kritériumok az optimális élsorozatokra

Tegyük fel egyenlőre, hogy az előző pontban leirt algoritmus segitségével az input kép összes csomópontját sikerült megkapnunk. Ennek fennállását a következő pont végén ellenőrizzük, és a hibás esetek kezelését is ott irjuk le.

Az eddigiekben megkaptuk az input képből az élmátrixot és a sulymátrixot, és az élmátrix bizonyos elemeit kijelöltük csomópontoknak. Következő feladatunk az, hogy keressünk a csomópontok között olyan élsorozatokat, amelyek a legjobban közelitik a kép konturvonalait, méghozzá ugy, hogy minden konturvonalat közelitsen élsorozat, és mindegyiket csak egy. Ebben a pontban az ilyen élsorozatokat jellemezzük néhány matematikai kritériummal.

Transzformáljuk a sulymátrix minden elemét ugy, hogy az uj sulyérték ε_1 és ε_2 közé essen / $\varepsilon_1 > 0$ és $\varepsilon_2 > \varepsilon_1$/ és az uj sulyérték annál kisebb legyen, minél "biztosabb" az él. A régi " suly alapján az uj W' érték lehet pl.

$$W' = \varepsilon_1 W + \varepsilon_2 (1-W)$$

Az optimális élsorozatokat a csomópontok közötti, később definiálandó értelemben minimális összsulyu utakként keressük az élmátrixban. Az $\varepsilon_1 > 0$ feltétel azért szükséges, hogy a minimális összsulyu élsorozatok a legrövidebbek is legyenek, tehát minél kevesebb kitérővel kövessék a konturt. ε_1 és ε_2 aránya azt fejezi ki, hogy az élsorozat rövidsége illetve az azokat alkotó élek "jósága" milyen sullyal essen a latba az optimális élsorozatok összeállitásánál.

Vezessük be a következő jelöléseket és definiciókat: Legyen G egy gráf, amelynek csucsai az élmátrix azon elemei, ahol sikerült élet találni. G-ben legyenek élek azok közt a csucsok között, amelyek az élmátrixban szomszédosak voltak, tehát amelyek egymást átfedő ablakokból származnak. Legyenek $\alpha_1, \alpha_2, \ldots, \alpha_c$ G-nek azok a csucsai, amelyeket csomópontoknak megjelöltünk. A G gráf csucsait általában β és γ betükkel fogjuk jelölni. Minden $\beta \in G$ csucshoz hozzá van rendelve a sulymátrix megfelelő W_β eleme, ezt a csucs sulyának, vagy költségének nevezzük.

Az élsorozatok a gráfban két csomópont közötti $(\alpha_1, \beta_1, \beta_2, \ldots, \beta_n, \alpha_j)$ utak lesznek.

Definició: Egy G-beli $U = (\beta_1, \ldots, \beta_n)$ ut költségének nevezzük a $c(U) = \beta_2 + \ldots + \beta_{n-1}$ mennyiséget. G két csucsa, β_1 és β_2 között a minimális költségü ut költségét $k(\beta_1, \beta_2)$-vel jelöljük.

Természetesen a β_1 és β_2 pontok között G-ben több minimális költségü ut is előfordulhat.

Definició: Legyen α_i és α_j két csomópont β pedig egy tetszőleges pont G-ben. Azt mondjuk, hogy β közelebb van α_i-hez, mint α_j-hez, ha $k(\alpha_i,\beta) < k(\alpha_j,\beta)$, vagy $k(\alpha_i,\beta) = k(\alpha_j,\beta)$ esetén, ha $i < j$. A minimális költségek között ezt a relációt jelöljük $k(\alpha_i,\beta) \overset{\bullet}{<} k(\alpha_j,\beta)$-val.

A $\overset{\bullet}{<}$ reláció arra szolgál, hogy ha két szóbajövő ut egyenlő költségü, akkor is egyértelmüen tudjunk minimális költségüt választani.

Definició: Legyen α_i és α_j két csomópont G-ben, és $i < j$.

Az $U = (\alpha_i, \beta_1, \ldots, \beta_n, \alpha_j)$ ut felezőpontjainak nevezzük a β_m és β_{m+1} pontokat, ha

$$W_{\beta_1} + \ldots + W_{\beta_{m-1}} < W_{\beta_{m+1}} + \ldots + W_{\beta_n}$$

és

$$W_{\beta_1} + \ldots + W_{\beta_m} \geqslant W_{\beta_{m+2}} + \ldots W_{\beta_n}$$

fennáll.

Definició: Az U_1 és U_2 utakat nem lényegesen különbözőknek nevezzük, ha két végpontjuk azonos, $c(U_1) = c(U_2)$ és a két utnak van közös felezőpontja.

Definició: Az U és V utakat ekvivalensnek nevezzük /jele: $U \approx V$/, ha találhatók olyan $U_1, U_2, \ldots, U_n$ utak, hogy $U_o = U$, $U_{n+1} = V$ jelöléssel minden $0 \leqslant i \leqslant n$-re U_i nem lényegesen különböző U_{i+1}-től.

Azonnal látható, hogy egy ekvivalencia-relációt definiáltunk a G-beli utak között.

Most már megfogalmazhatjuk a keresett $(\alpha_i, \beta_1, \ldots, \beta_n, \alpha_j)$ élsorozatokra vonatkozó kritériumokat.

a/ kritérium: Minden $1 \leqslant s \leqslant n$-re

$$\min \{k(\alpha_i, \beta_s), k(\alpha_j, \beta_s)\} \overset{\circ}{<} \min_{t \neq i,j} \{k(\alpha_t, \beta_s)\}$$

álljon.

Ez a kritérium azt fejezi ki, hogy egy élsorozat nem haladhat más csomópontok közelében. Pl. a 3.2.1. ábrán az α_1 és α_3 csomópontok között nyilván nem akarunk élsorozatot találni, noha az α_1 és α_3 közötti minimális költségü ut nem halad át α_2-n. Szeretnénk viszont élsorozatot találni α_1 és α_2 ill α_2 és α_3 között.

b/ kritérium: Minden $1 \leqslant s \leqslant n$-re

$$\min \{k(\alpha_i, \beta_s), k(\alpha_j, \beta_s)\} = \min \{W_{\beta_1} + \ldots + W_{\beta_{s-1}}, W_{\beta_{s+1}} + \ldots + W_{\beta_n}\}$$

álljon.

Ez a kritérium kizárja, hogy az ut valamelyik pontjából az utat "olcsóbban" be lehessen fejezni. Ez jelenti azt, hogy az ut minden pontja optimálisan kövesse a konturt.

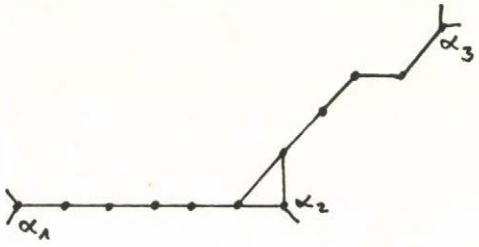
c/ kritérium: Az ut β_m és β_{m+1} felezőpontjaira a $\overset{\circ}{<}$ reláció szerint

$$\min_{t \neq i} \{k(\alpha_t, \beta_m)\} = k(\alpha_i, \beta_m)$$

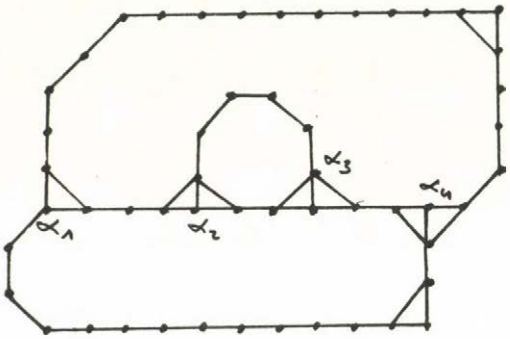
álljon, és a β_m-től α_j-hez vezető minimális költségü utak közül

legalább egy haladjon át β_{m+1}-en. Ugyan ez álljon fenn i és j, valamint β_m és β_{m+1} közötti szerepcserével is.

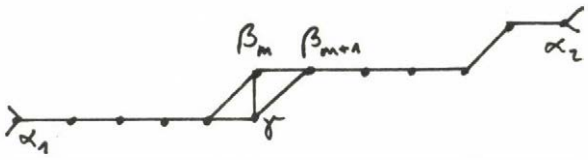
Ez a kritérium a b/ kritérium kiegészitése. A b/ kritérium ugyanis a felezőpontoknál nem kényszeriti ki, hogy az ut minimális költségü legyen. Például a 3.2.2. ábrán a β_m és β_{m+1}-en áthaladó ut kielégiti a b/ kritériumot, holott a γ-n át haladó ut kisebb költségü lehet. /A 3.2.1-3.2.3. ábrán minden csucs sulyát egyenlőnek vesszük./



3.2.1. ábra



3.2.3. ábra



3.2.2. ábra

d/ kritérium: Minden olyan ekvivalencia-osztályból, amelyik tartalmaz az a/, b/, c/ kritériumoknak eleget tevő utat, pontosan egy ut tartozzon az élsorozatok közé.

Ez a kritérium biztositja azt, hogy a kép minden konturvonalát kövesse élsorozat, és mindegyiket csak egy.

Természetesen előfordulhat, hogy két csomópont között több különböző élsorozatot is ki kell választanunk és lehet, hogy ezek közül egyik sem a minimális költségü ut a két csomópont között. Ilyen eset látható pl. a 3.2.3. ábrán.

A számunkra szükséges élsorozatokat legjobban felezőpontjaikkal tudjuk megfogni, ez az oka annak, hogy kritériumainkban ilyen sokat szerepelnek az utak felezőpontjai.

Célunk az, hogy előállitsuk élsorozatoknak egy olyan rendszerét, amely eleget tesz az a/-d/ kritériumoknak. A következő pontban erre a feladatra mutatunk be egy kvázi-párhuzamos algoritmust.

3.3. Egy kvázi-párhuzamos algoritmus optimális élsorozatok megkeresésére

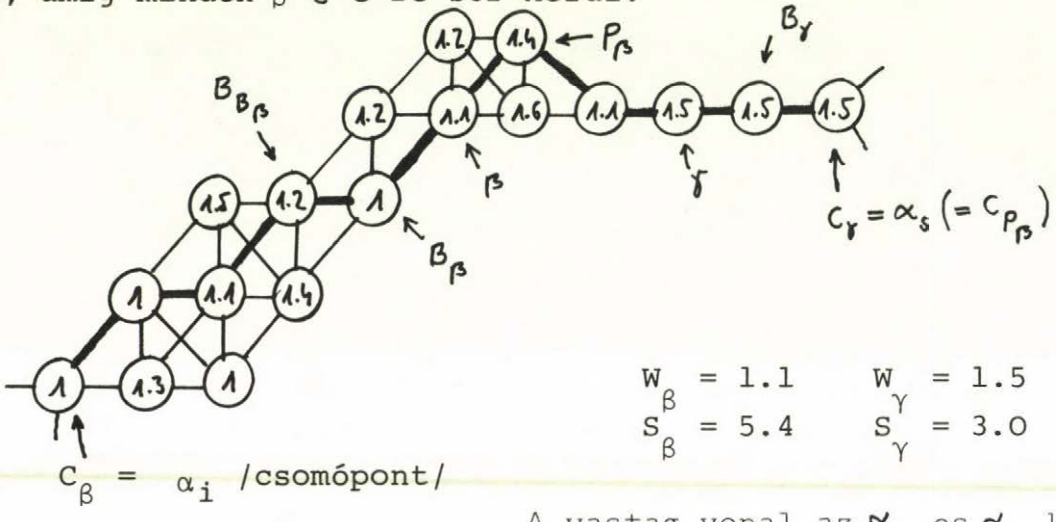
Az algoritmus lényegében egy potenciálmezőt növeszt a csomópontok körül a csucsok sulyértékei szerint. A keresett élsorozatok felezőpontjait azokban a pontokban találjuk meg, ahol két, különböző csomópontból eredő potenciálérték szomszédos, és a két potenciálérték összegének lokális minimuma van. Ezután a felezőpontokból visszakövetve a potenciálmező növekedését megkaphatjuk a keresett utakat a gráfban.

A G gráf minden β csucsához jelenleg egy W_β sulyérték van rendelve. Az algoritmus során minden csucshoz további négy számot, C_β, S_β, B_β és P_β-t fogunk rendelni. Kezdetben legyen minden α_i csomópontnál $C_{\alpha_i} = \alpha_i$, $B_{\alpha_i} = S_{\alpha_i} = P_{\alpha_i} = O$, G minden más csucsánál pedig legyen $C_\beta = B_\beta = S_\beta = P_\beta = O$. A C_β, B_β és P_β számok pointerek lesznek a G valamelyik másik csucsára /itt tegyük fel, hogy a O értelmetlen pointerérték/, méghozzá C_β pointer lesz arra a csomópontra, ahonnan kiindulva a potenciálmező először eléri a β csucsot, B_β G-nek arra a csucsára mutat, amelyen keresztül β-hoz jutott a potenciálmező, P_β pedig a felezőpontoknál fog az ut másik felezőpontjára mutatni. Az S_β szám a potenciálmező β-beli értékét tartalmazza /ld. 3.3.1. ábra/.

Legyen K az algoritmus során egy küszöbszám, kezdetben legyen $K = \varepsilon_1$. Az algoritmus minden lépése után K értékét ε_1-gyel megnöveljük.

Az algoritmus minden lépésben minden olyan $\beta \in G$-re, amelyre még $C_\beta = O$ és van olyan β' szomszédja, amelyre $S_\beta, +W_\beta < K$ és $C_{\beta'} \neq O$, legyen $C_\beta = C_{\beta'}$; $S_\beta = S_{\beta'} + W_\beta$ és $B_\beta = \beta'$ egy olyan

β' szomszéddal, amelyre S_β, minimális. Ha több ilyen β' is van, akkor egy olyan β'-t kell választanunk, amelynél $\alpha_i = C_\beta$, -re i minimális. A P_β értékeket egyenlőre nem definiáljuk. Ha minden $\beta \in G$-re, amelyre lehet, elvégeztük a fentieket, akkor növeljük K értékét ε_1-gyel és végezzük el ujra ezt az eljárást mindaddig, amig minden $\beta \in G$-re sor kerül.



$W_\beta = 1.1$ $W_\gamma = 1.5$
$S_\beta = 5.4$ $S_\gamma = 3.0$

A vastag vonal az α_i es α_s közötti keresett ut. Felezőpontjai: β és P_β

3.3.1. ábra

A következő három lemma az eddig leirt algoritmus és az azáltal létrehozott struktura néhány tulajdonságára mutat rá.

1. lemma. Ha β, $\beta' \in G$ és $S_\beta < S_{\beta'}$, akkor $S_{\beta'}$ nem kaphatta meg értékét az algoritmus korábbi lépésében, mint S_β.

Bizonyitás: Azt bizonyitjuk, hogy az n-edik lépésben kapott $S_\beta^{(n)}$ értékekre

$$(n - 1)\,\varepsilon_1 < S_\beta^{(n)} \leq n\varepsilon_1$$

és hogy minden, ennek az egyenlőtlenségnek eleget tevő $S_\beta^{(n)}$ értéket a n-edik lépésben meg is kapunk. Ebből a lemma következik.

Ezt az állitást teljes indukcióval bizonyitjuk. n = 1-re az állitás triviális. Tegyük fel, hogy minden i < n-re igaz és tekintsünk egy, az (n+1)-edik lépésben beirt $S_\beta^{(n+1)}$ értéket. Egy-

részt az algoritmus definiciója szerint $S_\beta^{(n+1)} \leq (n+1)\varepsilon_1$, és az indukciós feltevés szerint, ha $S_\beta^{(n+1)}$-et csak az (n+1)-edik lépésben irtuk be, akkor $S_\beta^{(n+1)} > n\varepsilon_1$. Másrészt, ha egy β pontra $n\varepsilon_1 < S_\beta < (n+1)\varepsilon_1$, akkor mivel valamelyik β' szomszédjára $S_\beta = S_{\beta'} + W_\beta$ és $W_\beta > \varepsilon_1$, $S_{\beta'}$-t az indukciós feltevés szerint legkésőbb az n-edik lépésben megkaptuk, igy az algoritmus definiciója szerint S_β-t is meghatároztuk az (n+1)-edik lépéssel. Ezzel az 1. lemmát bebizonyitottuk. ■

2. lemma. Ez a potenciálmező-növesztő eljárás kvázi-párhuzamos, azaz minden lépésben a β csucsok tetszőleges sorrendben, akár /részben vagy teljesen/ egyidejüleg is megkaphatják C_β, S_β és B_β értékeiket, az eredményt ettől nem változik.

Bizonyitás: Mivel minden $\beta \in G$-re $W_\beta > \varepsilon_1$, az 1. lemma bizonyitásához felhasznált állitásból következik, hogy az algoritmus egyik lépésében sem fordulhat elő az, hogy valamelyik β csucs olyan β' csucstól kapja C_β, S_β és B_β értékét, amelyik ugyanabban a lépésben kapta értékeit. /Ezért kell a K-t csak ε_1-enként növelni./ Ebből a kvázi-párhuzamosság következik. ■

3. lemma. Minden $\beta \in G$-re ami nem csomópont, az összes csomópontból a β-ba vezető utak közül a $\overset{\circ}{<}$ reláció szerint egy minimális költségü a

$$\beta,\ B_\beta,\ B_{B_\beta},\ \ldots,\ B_{B_{B\ldots B_\beta}} = C_\beta$$

ut, amelynek költsége $S_\beta - W_\beta$.

Bizonyitás: Tegyük fel, hogy az állitás nem igaz, és válasszunk ki azok közül a β-k közül, melyek ellenpéldák az állitásra egy olyat, amelyre S_β minimális.

Ha a csomópontokból β-hoz vezető utak közül egy minimális költségü B_β-n át vezet, akkor már B_β-hoz is el lehet jutni

$S_{B_\beta} = S_\beta - W_\beta < S_\beta$ költséggel, ami ellentmond S_β minimális voltának.

Ha pedig csak β-nak $\beta' \neq B_\beta$ szomszédján át vezet β-hoz csomópontból vezető, minimális költségü ut, akkor $S_{\beta'} < S_{B_\beta} = S_\beta - W_\beta < S_\beta - \varepsilon_i$ miatt az 1. lemma szerint, amikor S_{B_β} értéket kapott, legkésőbb abban a lépésben $S_{\beta'}$ is, és S_β addig még nem kapott értéket. Eszerint viszont B_β értéke ellentmond az algoritmusnak, hiszen S_β-t nem S_{B_β}, hanem pl. $S_{\beta'}$ alapján kellett volna meghatározni. Ezzel a lemma bizonyitását befejeztük. ■

Most kitöltjük a P_β értéket is. Minden $\beta \in G$-re legyen $P_\beta = O$, ha β minden β' szomszédjára $C_{\beta'} = C_\beta$. Egyébként P_β legyen az a β' szomszédja β-nak, amelyre $S_{\beta'}$ minimális, és $C_{\beta'} \neq C_\beta$.

Ha több ilyen β' is van, akkor válasszuk azt, amelyiknél $\alpha_i = C_{\beta'}$-re i minimális. Ha ilyen β' is több van, akkor ezek közül azt válasszuk, amelyik leginkább balra, és ezek közül azt, amelyik leglejjebb van az élmátrixban. Természetesen a P_β értékek meghatározása is végezhető kvázi-párhuzamosan, mert a P_β-k egymástól függetlenek.

4. lemma. Ha egy $\beta \in G$ pontra $P_\beta \neq O$, akkor az $\alpha_i = C_{P_\beta}$ és $\alpha_j = C_\beta$ csomópontok közötti alábbi U_β utra:

$$U_\beta = (C_{P_\beta} = B_{B\cdots B_{P_\beta}}, \ldots, B_{B_{P_\beta}}, B_{P_\beta}, P_\beta, \beta, B_\beta, B_{B_\beta}, \ldots, B_{B\cdots B_\beta} = C_\beta)$$

teljesül az a/ és b/ kritérium.

Bizonyitás: Az algoritmus konstrukciójából következik, hogy az U_β ut minden β előtti β' pontjára $C_{\beta'} = C_{P_\beta}$, és minden β utáni β' pontjára $C_{\beta'} = C_\beta$. Ebből a 3. lemma szerint az a/ kritérium következik. A b/ kritérium fennállása szintén a 3. lemma egyenes következménye. ■

5. lemma. Ha egy α_i, β_1, ..., β_n, α_j ut eleget tesz az a/ b/ kritériumoknak, akkor az ut bármelyik β_m felezőpontjára $P_{\beta_m} \neq O$.

Bizonyitás: Az a/ kritériumból a 3. lemma miatt következik, hogy minden $1 \leqslant q \leqslant n$-re vagy $C_{\beta_q} = \alpha_i$, vagy $C_{\beta_q} = \alpha_j$. Tegyük fel például, hogy $C_{\beta_m} = \alpha_j$. /Ha $C_{\beta_m} = \alpha_i$, akkor ugyanezt a bizonyitást az ut másik felére kell elmondani./

Tegyük fel, hogy a lemma nem igaz, és $P_{\beta_m} = O$. A P_β-értékek definiciójából következik, hogy ekkor $C_{\beta_{m-1}} = \alpha_j$. A 3. lemma szerint ebből következik, hogy $W_{\beta_1} + \ldots + W_{\beta_{m-2}} \geqslant k(\alpha_j, \beta_{m-1})$, mivel $\beta_1, \ldots, \beta_{m-2}$ egy ut α_i és β_{m-1} között és mégis $C_{\beta_{m-1}} = \alpha_j$. Mivel azonban β_m az ut felezőpontja,

$$W_{\beta_{m+1}} + \ldots + W_{\beta_n} \geqslant W_{\beta_1} + \ldots + W_{\beta_{m-1}} > W_{\beta_1} + \ldots + W_{\beta_{m-2}} \geqslant k(\alpha_j, \beta_{m-1})$$

ami ellentmond a b/ kritériumnak a β_{m-1} pontban, hiszen ha $C_{\beta_{m-1}} = \alpha_j$, akkor az algoritmus definiciója miatt $W_{\beta_1} + \ldots + W_{\beta_{m-1}} \geqslant W_{\beta_{m+1}} + \ldots + W_{\beta_n}$ is fennáll. Ezzel a lemmát bebizonyitottuk. ∎

6. lemma. Ha egy $\beta \in G$ pontra $P_\beta \neq O$ és $P_{P_\beta} = \beta$, akkor a β-n keresztül a 4. lemmában leirt módon konstruált U_β ut felezőpontjai β és P_β , és az ut eleget tesz a c/ kritériumnak is.

Bizonyitás: Jelöljük az U_β ut elemeit C_β, β_1, ..., β_n, C_{P_β}-val, ahol $\beta = \beta_m$ és $P_\beta = \beta_{m+1}$. Az U_β ut definiciójából következik, hogy az S_{β_q} értékek $q = m$ -ig szigoruan nőnek, és $q = (m+1)$-től szigoruan csökkennek. Ezért, mivel egyenlőség esetén mind a felezőpont definiciójánál, mind a potenciálmező növesztésénél a csomópontok lexikografikus sorrendje döntött, U_β felezőpontjai

csak β_m és β_{m+1} lehetnek. Ezekre azonban a P_β értékek definiciójából következik a c/ kritérium a 2. lemma miatt.■

Készitsük most el minden olyan β, β' pontpárra, amelyre $P_\beta = \beta'$ és $P_{\beta'} = \beta$ a 4. lemmában leirt módon az $U_\beta = U_{\beta'}$ utat. Igy G-beli utaknak egy $\mathcal{U}$ rendszerét kapjuk, amire igaz a következő tétel:

Tétel: $\mathcal{U}$ minden eleme eleget tesz az a/, b/ és c/ kritériumoknak, és minden olyan G-beli U uthoz, amelyik eleget tesz az a/, b/, c/ kritériumoknak található olyan $U' \in \mathcal{U}$, hogy U' ekvivalens U-val.

Bizonyitás: A 3. és 5. lemmából következik, hogy minden $\mathcal{U}$-beli ut eleget tesz az a/, b/ és c/ feltételeknek. Két β, $\beta' \in G$ csucs között a $\prec$ relációt értelmezzük a következőképpen: Ha β az élmátrix x_1-edik sorában az y_1-edik elem, β' pedig az élmátrix (x_2,y_2) elemének felel meg a gráfban, akkor $\beta \prec \beta'$ ha $x_1 < x_2$ vagy $x_1 = x_2$ esetén, ha $y_1 < y_2$. Tegyük fel, hogy az $U_o = (\alpha_i, \beta_1^o, \ldots, \beta_{n_o}^o, \alpha_j)$ ut eleget tesz az a/, b/, c/ kritériumoknak, és U_o felezőpontjai legyenek $\beta_{m_o}^o$ és $\beta_{m_o+1}^o$. Az 5. lemma szerint tehát $P_{\beta_{m_o}^o} \neq O$.

Jelöljük a $\beta_{m_o}^o$ -on keresztül a 4. lemmában leirt módon konstruált utat $U_1 = (\alpha_i, \beta_1^1, \ldots, \beta_{n_1}^1, \alpha_j)$ -vel. Az U_o-ra fennálló c/ kritérium és a P_β -értékek definiciója miatt ennek az utnak is az α_j csomópont a végpontja, és U_1 nem lényegesen különbözik U_o-tól. Legyenek U_1 felezőpontjai $\beta_{m_1}^1$ és $\beta_{m_1+1}^1$. A 6. lemma miatt $\beta_{m_1}^1 = \beta_{m_o}^o$, és a P_β értékek definiciója szerint $\beta_{m_1+1}^1 \prec \beta_{m_o+1}^o$.

A 4. és 6. lemma szerint U_1 is kielégiti az a/, b/, c/ kritériumokat, ezért az 5. lemma szerint erre is $P_{\beta_{m_1+1}^1} \neq O$, igy $\beta_{m_1+1}^1$-en keresztül megkonstruálhatjuk a 4. lemma szerinti

$U_2 = (\alpha_i, \beta^2_1, \ldots, \beta^2_{n_2}, \alpha_j)$ utat. Ennek $\beta^2_{m_2}$ és $\beta^2_{m_2+1}$ felezőpontjaira $\beta^2_{m_2} \prec \beta^1_{m_1}$ és $P_{\beta^2_{m_2}} \neq O$ áll, és U_2 sem különbözik lényegesen U_1-től. Most $\beta^2_{m_2}$-n keresztül az eddigiekhez hasonlóan megkonstruáljuk az U_3 majd az U_4, U_5 ... utakat. Minden i-re U_i nem lényegesen különbözik U_{i+1}-től, igy ha megmutatjuk, hogy található olyan j, amelyre $U_j \in \mathcal{U}$, akkor tételünket bebizonyitottuk. /Akkor persze $U_j = U_{j+1} = \ldots$ is fennáll ettől a j-től./

Az U_i utak felezőpontjaira sorra fennállnak a következő relációk:

$$\beta^o_{m_o} = \beta^1_{m_1} \prec \beta^2_{m_2} = \beta^3_{m_3} \prec \beta^4_{m_4} \quad \ldots \quad \text{és}$$

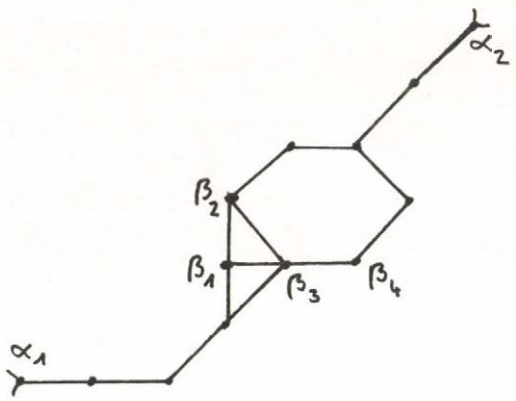
$$\beta^o_{m_o+1} \prec \beta^1_{m_1+1} = \beta^2_{m_2+1} \prec \beta^3_{m_3+1} = \beta^4_{m_4+1} \quad \ldots$$

Mivel G véges, és a $\prec$ reláció olyan, hogy ha $\beta \neq \gamma$, akkor vagy $\beta \prec \gamma$ vagy $\gamma \prec \beta$ fennáll, mindkét sorban található olyan r ill. s index, hogy attól kezdve mindig a $\prec$ jelek helyén is = áll. Jelölje t r és s közül a nagyobbikat. Azt kaptuk tehát, hogy van olyan t szám, amire $\beta^t_{m_t+1} = \beta^{t+1}_{m_{t+1}+1}$ és $\beta^t_{m_t} = \beta^{t+1}_{m_{t+1}}$, ami azt jelenti, hogy az U_t ut definiciója szerint, hogy U_t $\beta^t_{m_t}$, $\beta^{t+1}_{m_t+1}$ felezőpontjaira $P_{P_{\beta^t_{m_t}}} = \beta^t_{m_t}$ fennáll, azaz $U_t \in \mathcal{U}$. Ezzel a tétel bizonyitását befejeztük. ■

A tétel eredményeképpen tehát azt kaptuk, hogy ha leirt módon megkonstruáljuk az a/, b/, c/ kritériumoknak eleget tevő utak rendszerét, akkor $\mathcal{U}$-ban minden ekvivalencia-osztály képviselve van. Sajnos azoban az nem feltétlenül igaz, hogy minden ekvivalencia-osztályt csak egy ut képvisel $\mathcal{U}$-ban. Erre a 3.3.2. ábrán láthatunk egy ellenpéldát. Legyen az ábrán látható gráf minden csucsához ugyanaz a W sulyérték rendelve. Az α_1 és α_2 csomópontokat β_1 és β_2 felezőpontokon át, ill. a β_3 és β_4 felezőpontokon át összekötő utak ekvivalensek, de mindkettő szerepel -ban, mert $P_\beta = \beta_2$, $P_{\beta_2} = \beta_1$, $P_{\beta_3} = \beta_4$ és $P_{\beta_4} = \beta_3$ fennáll.

A következő algoritmus segitségével kiszürjük $\mathcal{U}$-ból az egymással ekvivalens utakat, amivel kitüzött célunkat elérjük.

Az első olyan β, β' párra, amelyre $P_\beta = \beta'$ és $P_{\beta'} = \beta$ minden olyan β_1 szomszédját, amelyre $C_{\beta_1} = C_\beta$, és $S_{\beta_1} = S_\beta$, jelöljük meg. Az igy megjelölt csucsoknak minden olyan β_2 szomszédját is jelöljük meg, amelyre $C_{\beta_2} = C_\beta$ és $S_{\beta_2} = S_\beta$. Az utoljára megjelölt csucsoknak ismét jelöljük meg azokat a β_3 szomszédjait, melyekre $C_{\beta_3} = C_{\beta'}$ és $S_{\beta_3} = S_{\beta'}$. Folytassuk ezt a megjelölési eljárást mindaddig, amig találunk uj megjelölendő csucsot. A C és S értékek összehasonlitásánál a β és β' értékek mindig felváltva következzenek. Ha a megjelölt csucsok között



3.3.2. ábra

van olyan γ_1, γ_2 csucs, amelyre $P_{\gamma_1} = \gamma_2$ és $P_{\gamma_2} = \gamma_1$, akkor $U_\beta \approx U_{\gamma_1}$, mert a megjelölési algoritmus szerint U_β-tól eljuthatunk egymástól lényegesen nem különböző utak sorozatával U_{γ_1}-hez. Ezért az ilyen U_γ utakat nyugodtan elhagyhatjuk $\mathcal{U}$-ból, mert maradt velük ekvivalens ut $\mathcal{U}$-ban /ugyanis U /.

Másrészt az ekvivalencia definiciójából következik, hogy ez az algoritmus $\mathcal{U}$-ból minden U_β-val ekvivalens utat eltüntet.

A megmaradt β, β' párokra, ahol $P_\beta = \beta'$ és $P_{\beta'} = \beta$ szintén sorra végezzük el a fenti megjelölési és törlési algoritmust. Az igy megmaradt $\mathcal{U}$ rendszer már valóban minden ekvivalencia-osztályból csak egy utat tartalmazhat, igy ezzel célunkat elértük.

A megjelölési és törlési algoritmus standard backtrackelési eljárással implementálható. Ez az algoritmus ugyan sorosan végezendő, viszont nagyon gyorsan lefut, mint azt a következő gondolatmenet is mutatja:

Minden $\gamma \in G$ csak olyan β, β' párok kapcsán jelölődhet meg, melyekre γ valamelyik γ' szomszédjával $C_\gamma = C_\beta$ és $C_{\gamma'} = C_{\beta'}$, vagy $C_\gamma = C_{\beta'}$, és $C_\beta = C_{\gamma'}$, fennáll.

Mivel az U_β uttal ekvivalens utak a megjelölési eljárásnak csak egyetlen lépésében szerepelnek, a 2. lemmából és a b/ kritériumból következik, hogy γ minden szomszédja révén legfeljebb egyszer kerülhet megjelölésre. Mivel G minden csucsának legfeljebb 8 szomszéda van, és minden megjelölés után azonnal ellenőrizhető, hogy ezáltal vált-e szükségessé törlés, ez az egész algoritmus o(N) lépésben lefut, ha a gráfnak N csucsa van. Valójában a sulyértékek különbözősége miatt kevés ekvivalens ut van G-ben, és a pontok többsége sem felezőpont, igy az egész megjelölési és törlési algoritmus általában nagyon gyorsan /néhány ezredmásodperc alatt/ lefut.

A tétel alapján és az imént leirt algoritmus segitségével előállithatjuk a kivánt élsorozatokat a csomópontok között. Könnyen belálható, hogy az egész algoritmus k processzor alkalmazása esetén $O(\frac{N^2}{k})$ lépést igényel. N processzor használata esetén tehát algoritmusunk O(N) lépést végez. k = 1 processzor esetén a kisérletekben kapott futási időket a 4. fejezetben tárgyaljuk.

Ezután feladatunk már csak annyi, hogy eldöntsük, hogy valóban megtalálta-e a csomópontkereső algoritmus az ábra összes csomópontját, vagy kell még további csomópontokat detektálni.

További csomópontok detektálására a következő esetekben van szükség:

1/ Ha két nem ekvivalens élsorozat hosszabb darabon együtt halad, akkor csomópontot kell jelezni ott, ahol egymástól elágaznak.

2/ Ha két ekvivalens ut a gráf valamely részén egymástól távol halad, akkor a külön nem észlelt utnak egy, a másik uttól távoli pontját külön ki kell jelölni csomópontnak, hogy ezt az utat is észlelhessük. /Ez a csomópont a látványgráfban másodfoku csucs lesz./

3/ Ha a készitendő látványgráfban egy körutnak legfeljebb egy legalább harmadfoku csucsa van, ezt az utat is ujabb /később másodfokuvá váló/ csomópont ideiglenes beillesztésével kell észlelhetővé tennünk.

4/ Detektálni kell a látványgráf esetleges elsőfoku csucsait és az odavezető élsorozatokat is.

Algoritmusunk további lépéseiben ezeket a feladatokat oldjuk meg. A lényegesen különböző, a/, b/, c/ kritériumoknak eleget tevő utaknak a tétel alapján való meghatározása során G minden pontjáról jelöljük meg, hogy pontja-e valamelyik észlelt utnak. Külön jellel /J/ jelöljük meg azokat a pontokat, amelyek több talált utnak is pontjai. /Ezeknél jön szóba az 1. eset/.

Legyen T egy előre meghatározott küszöbszám. T fejezze ki azt a türési határt, ameddig nem tartjuk zavarónak azt, hogy két ut egy legfeljebb T költségü szakaszon együtt fusson, vagy azt, hogy egy észlelt uttól legfeljebb T költségnyire lévő élek ne tartozzanak egy uthoz sem.

Definició: A β pont és β' szomszédja S-értékeire azt mondjuk, hogy $S_\beta \overset{\circ\circ}{<} S_{\beta'}$, ha $S_\beta < S_{\beta'}$, vagy egyenlőség esetén, ha β az élmátrixban β'-től jobbra vagy, itt is egyenlőség esetén, fölfelé van.

Most ujabb csomópontot detektálunk azokban a pontokban, ahol J áll, $S_\beta > T$, és β minden β' szomszédjára, ahol J áll, $S_{\beta'} \overset{\circ\circ}{<} S_\beta$.

A továbbiakban még azt kell megvizsgálni, hogy melyek azok a pontok, amelyek minden olyan ponttól, amelyeket valamelyik utnál felhasználtunk, T-nél nagyobb távolságra vannak. E célból vébezzük el potenciálmező-növesztő eljárásunkat /ismét kvázi-párhuzamos módon/ ugy, hogy a csomópontok szerepét /aholis $S_\beta = O$/ most az összes olyan pont játssza, amelyet valamelyik utnál felhasználhatunk. Ezután ujabb csomópontokat detektálunk azokban a $\beta \in G$ pontokban, ahol $S_\beta > T$ és β minden β' szomszédjára $S_{\beta'} \overset{\circ\circ}{<} S_\beta$.

Az eredeti és az ujonnan detektált csomópontokat vegyük ezután α_i-knek, és ismételjük a leirt élsorozat-kereső eljárást addig, amig uj csomópontokat már nem tudunk detektálni. Az algoritmus véges sok iteráció után véget ér, mivel a csomópontok száma minden lépésben nő. Többnyire legkésőbb a második ismétlés után megkapjuk az ábra összes csomópontját, és az őket összekötő kivánt élsorozatokat.

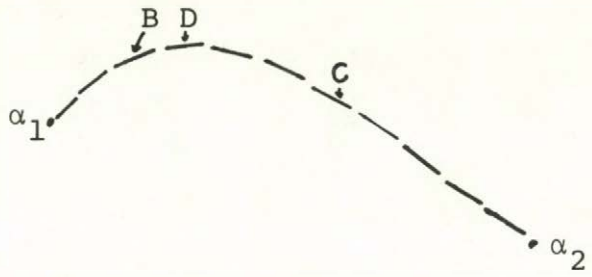
A számitógépes futtatásoknál az algoritmus paramétereinek értékét $\varepsilon_1 = 1$, $\varepsilon_2 = 2$, $T = 3{,}2$-nek választottuk.

3.4. Az optimális élsorozatok értelmezése

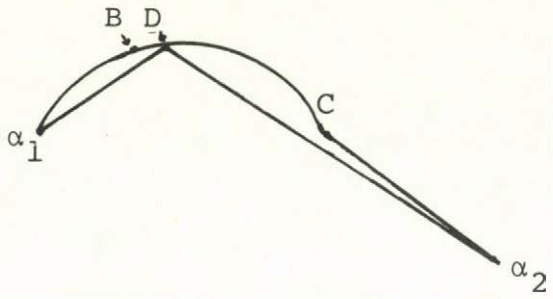
A 2.2. pontban leirtunk egy algoritmust, amely az élsorozatokból azokat közelitő egyenesszakaszokat és köriveket állit elő. Az az algoritmus, mint láttuk, a bizonytalanul értelmezhető élsorozatokra egyetlen egy értelmezést ad, és ezért az élsorozatban már egy-egy hibás élre is nagyon érzékeny tud lenni. Itt leirunk egy algoritmust, amely az élsorozatokra alternativ értelmezést is tud adni. Ez az algoritmus felhasznál a tárgyakról bizonyos á priori ismereteket /ilyen szempontból tehát "intelligensebb", mint a 2.2-beli/, és lehetőséget nyujt kvázi-párhuzamos feldolgozásra is.

Az algoritmus outputja a látványgráfnak egy olyan változata, mely a harmad- és magasabbfoku csucspontok között több, egymást alternativan helyettesitő lehetsége látványgráfbeli élsorozatot tartalmaz /az illető optimális élsorozat lehetséges értelmezéseként/. A 3.4.1. ábrán egy példa látható egy élsorozatra /α_1 és α_2 a 3.1. pontban talált és a 3.3. pont algoritmusa után is megmaradt csomópontok/; a 3.4.2. ábrán pedig ennek lehetséges értelmezései láthatók: vagy az α_1BC ivet és a Cα_2 egyenest, vagy az αD egyenest és a Dα_2 egyenest tekintjük eme élsorozat értelmezésének. /Természetesen a két értelmezés egymást kizárja/. Minden lehetséges értelmezéshez egy valószinüségi értéket is csatol az algoritmus, ez az érték azt fejezi ki, hogy a szóbanforgó élsorozatnak ez az ér-

telmezése mennyire valószinü a többi értelmezéshez képest. Minden élsorozatra a különböző értelmezések valószinüségeinek összege 1. A 3.4.2. ábrán pl. az első értelmezés valószinüségére 0,7, a másodikra 0,3 értéket kapunk. Ezek a valószinüség-értékek a későbbi felismerés döntéseit segitik.



3.4.1. ábra



3.4.2. ábra

Használjuk most is a 2.2. pont jelöléseit, tehát legyenek az egyenesdarabok sorra $\{s_1, \ldots, s_n\}$, α_k az s_k irányszöge, $\Delta_k = \alpha_{k+1} - \alpha_k \pmod{\Pi}$.

Az algoritmus először megkeresi az élsorozatban azokat az éleket, amelyeknél egyáltalán a konturvonalaknak töréspontja elképzelhető. Ehhez minden $3 \leq k \leq n-3$-ra legyen

$$d_{k,1} = \left| \frac{|\Delta_{k-2}| + |\Delta_{k-1}|}{2} - \frac{|\Delta_k| + |\Delta_{k+1}|}{2} \right|$$

$$d_{k,2} = \left| \frac{|\Delta_{k-2} - \Delta_{k-1}|}{2} - \frac{|\Delta_k - \Delta_{k+1}|}{2} \right|$$

$$d_k = \max \{ d_{k,1} , d_{k,2} \}$$

Legyenek $\{ i_m ; m=1, \ldots , p \}$ azok az indexek, amelyekre $d_{i_{m-1}} < d_{i_m} \leq d_{i_{m+1}}$. Ezek közül az i_m indexek közül válasszuk ki azt a N darabot, melyekre d_{i_m} a legnagyobb. /Ha $p < N$, akkor akkor mindegyiket kiválasztjuk/. Az igy kapott i_m-ekre azt mond-

juk, hogy az élsorozatban s_{i_m} -nél töréspont lehetséges. Mi legfeljebb N=8 töréspont lehetőséget veszünk figyelembe, mert N növekedtével a későbbi algoritmusok időszükséglete exponenciálisan nő. N=8 azonban általában bőségesen elégnek bizonyult. Az algoritmusok időszükségletét a 4. fejezetben fogjuk részletesen tárgyalni.

A további algoritmusban felhasználunk a felismerendő tárgyakról annyi á priori ismeretet, hogy tudhatjuk, hogy bármelyik látványgráfban két legalább harmadfoku csucspont között a kontur-vonalak legfeljebb öt különböző vonaldarabból /egyenesszakaszból vagy körivből/ állnak, és ezek közül is legfeljebb egy lehet körív. Ennek a ténynek az ismerete nagymértékben megkönnyiti az algoritmus tervezését és növeli biztonságát. Természetesen, ha uj tárgyakkal bővül a felismerendő tárgyak halmaza, akkor ez a feltételezés értelemszerüen módositható.

Ezek után már minden élsorozat értelmezésére csak a következő lehetőségek képzelhetők el: /L egyenesszakaszt, A pedig körivet jelent/:

a/ L	b/ A	c/ LL	d/ AL	e/ LA	f/ ALL
g/ LAL	h/ LLA	j/ ALLL	k/ LALL	l/ LLAL	m/ LLLA
n/ ALLLL	o/ LALLL	p/ LLALL	q/ LLLAL	r/ LLLLA.	

Az a/ és b/ eset 0,..., végül az n/-r/ esetek négy töréspontot feltételeznek a látványgráf eme élsorozatnak megfelelő részében. Az algoritmus ugy müködik, hogy külön-külön az a/-r/ esetek mindegyikéhez hozzárendel egy w_a, ..., w_r sulyértéket, ami azt fejezi ki, hogy az élsorozat mennyire felel meg az illető esetnek, a szükséges töréspontok optimális megválasztása esetén. A töréspontokat a d_{i_1}, ... , d_{i_N} töréspontlehetőségek közül választjuk, és ezek minden /szükség szerint 1-es, 2-es, 3-as vagy 4-es/ kombinációja esetén megvizsgáljuk, hogy a töréspontok közötti éldarabok mennyire felelnek meg az egyenesszakaszokkal ill. a körivekkel szemben támasztott kritériumoknak /2.2. pont/. Ezt a vizsgálatot két eljárás végzi, amelyknek inputja egy p_1, ... , p_k élsorozat,

/a teljes optimális élsorozat egy része/, outputja pedig egy-egy sulyérték, ami azt fejezi ki, hogy ez az élsorozat mennyire tekinthető egy egyenesnek, ill. egy körivnek. Ennek a két eljárásnak a müködését később részletesen ismertetjük. A w_a, ... , w_r értékeket ezeknek a sulyoknak az összegeként számoljuk, és végső értékük az összes lehetséges töréspontkombinációkból kapott értékek minimuma lesz.

Legyen most $W = \max\{w_a, \ldots, w_r\}$, K pedig egy előre adott küszöbszám. Ekkor az algoritmus az élsorozat értelmezéseként azokat az α alternativákat adja meg a a/-r/ lehetőségek közül, melyekre $w_\alpha > W - K$. Az ezekhez tartozó w_α sulyokból kapott $w_\alpha - W + K$ számokat 1-re normáljuk, igy kapjuk meg az egyes alternativákhoz rendelt "valószinüség-értékeket". Általában az algoritmus az egyes élsorozatokhoz 1-4 alternativ értelmezést ad. Ezek szinte mindig tartalmazzák az élsorozat helyes értelmezését is, olyankor is, amikor a 2.2. pontban leirt algoritmus téved. /A kisérleti eredményeket a 4. fejezetben elemezzük részletesen./

Hátra van még annak a két eljárásnak a bemutatása, amelyek egy p_1, ... , p_k élsorozathoz az egyenes- ill. köriv-értelmezésnek megfelelő sulyokat hozzárendelik.

Legyen most is α_1, ..., α_k a p_1, ... , p_k éldarabok irányszöge, továbbá $\Delta_i = \alpha_{i+1} - \alpha_i \pmod{\Pi}$. Mindkét eljárás alapgondolata az, hogy p_1, ... , p_k élek mindegyikéhez bizonyos mennyiségü jutalom, ill. büntetőpontot rendel, aszerint, hogy mennyire illeszkednek a többiekkel együtt egy egyenesre ill. ivre.

Definiáljuk a következő függvényeket: /Ezek rendelik az élekhez a jutalom- ill. büntetőpontokat/

$$l(x)=\begin{cases} 5, & \text{ha } x < 10^\circ \\ 3, & \text{ha } x < 15^\circ \\ 1, & \text{ha } x < 20^\circ \\ -1, & \text{ha } x < 25^\circ \\ -3, & \text{ha } x < 30^\circ \\ -5, & \text{ha } x > 30^\circ \end{cases} \qquad a(x)=\begin{cases} 5, & \text{ha } 15^\circ < x < 25^\circ \\ 3, & \text{ha } 7^\circ < x \text{ vagy } x < 30^\circ \\ 1, & \text{ha } 2^\circ < x \text{ vagy } x < 35^\circ \\ -1, & \text{ha } -2^\circ < x \text{ vagy } x < 45^\circ \\ -3, & \text{ha } -5^\circ < x \text{ vagy } x < 55^\circ \\ -5, & \text{ha } x \leqslant -5^\circ \text{ vagy } x \geqslant 55^\circ \end{cases}$$

Ezek után a $p_1, \ldots, p_k$ élsorozat egy egyenesként ill. körívként való értelmezéséhez rendelt L ill. A számok legyenek a következők:

$$L = \min_{i=1,\ldots,k} \left\{ \sum_{j=1}^{k} l(p_i - p_j) - 12 \right\}$$

$$A = \min \left\{ \sum_{j=1}^{k-1} a(p_{j+1} - p_j) - 12 \; , \; \sum_{j=1}^{k-1} a(p_j - p_{j+1}) - 12 \right\}$$

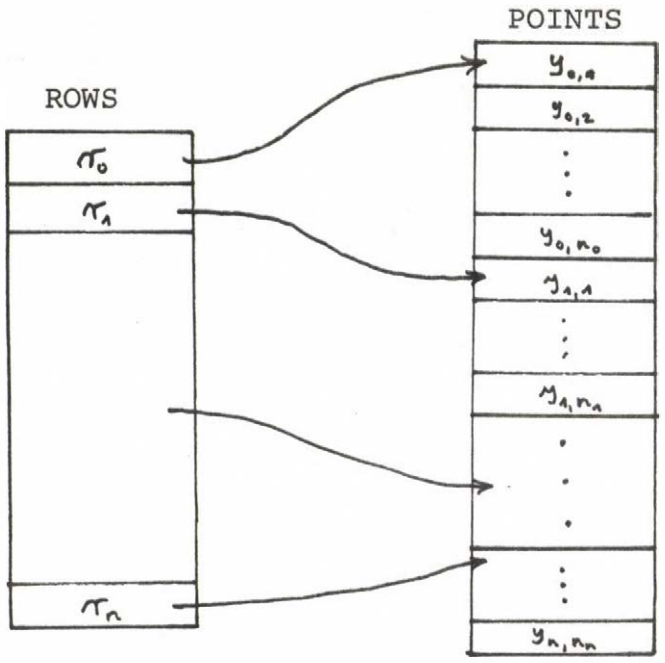
Mindkét esetben a 12 levonás azt fejezi ki, hogy mindig 12 büntetőpontot adunk pusztán azért, hogy egy vonalat elkezdhessünk. Igy érjük el azt, hogy pl. ha az egész optimális élsorozat egyetlen egyenest ábrázol, akkor a két egyenesként való értelmezés sulya lényegesen kisebb legyen. A kifejezésekben szereplő konkrét büntető- ill. jutalompont-értékek és küszöbszámok a kisérletezés során alakultak ki. Az L szám képletében a szumma értéke azt fejezi ki, hogy az egész élsorozat egy p_i irányának megfelelő egyenesnek mennyire tekinthető. Az A kiszámitásában pedig külön vizsgáljuk, hogy mennyire tekinthető az élsorozat konkáv ill. konvex ivnek. Az A számot csak $k > 3$ esetben vizsgáljuk, 4-nél kevesebb élet tartalmazó élsorozatot semmiképpen sem értelmezünk ivnek.

Látható, hogy ezt az algoritmust egy-egy hibás éldarab az élsorozatban lényegesen kevésbé zavarja, mint a 2.2. pontban bemutatott algoritmust. Részletes kisérleti eredmények a 4. fejezetben találhatók. Az egyes töréspontkombinációkhoz tartozó sulyértékek meghatározása természetesen kvázi-párhuzamosan is végezhető.

4. Implementáció, kisérleti eredmények, tapasztalatok

4.1. Az algoritmusok implementálása

A TV első üzemmódjából kapott képmátrixot sorfolytonosan tároljuk. A másik üzemmódból kapott képet ugy reprezentáljuk a számitógépben, hogy /n x m pontos képfelbontás esetén/ készitünk egy n szóból álló ROWS nevü listát és egy másik, POINTS nevü listát. A POINTS listában x,y szerinti lexikografikus sorrendben elhelyezzük a szürkeségi szintváltások helyeinek y-koordinátáit és az onnantól érvényes szürkeségi szintet. /4.1.1.ábra/ A ROWS lista minden a_i eleme egy pointer, amely a POINTS listában oda mutat, ahol az i-edik sorban lévő szürkeségi szint-változások felsorolása kezdődik. A lézer-képet ugyanilyen formátumban tároljuk, de a POINTS listában csupán az y-koordinátákat adjuk meg, hiszen a fényesség változásának mértékéről itt nincs információnk.



4.1.1. ábra

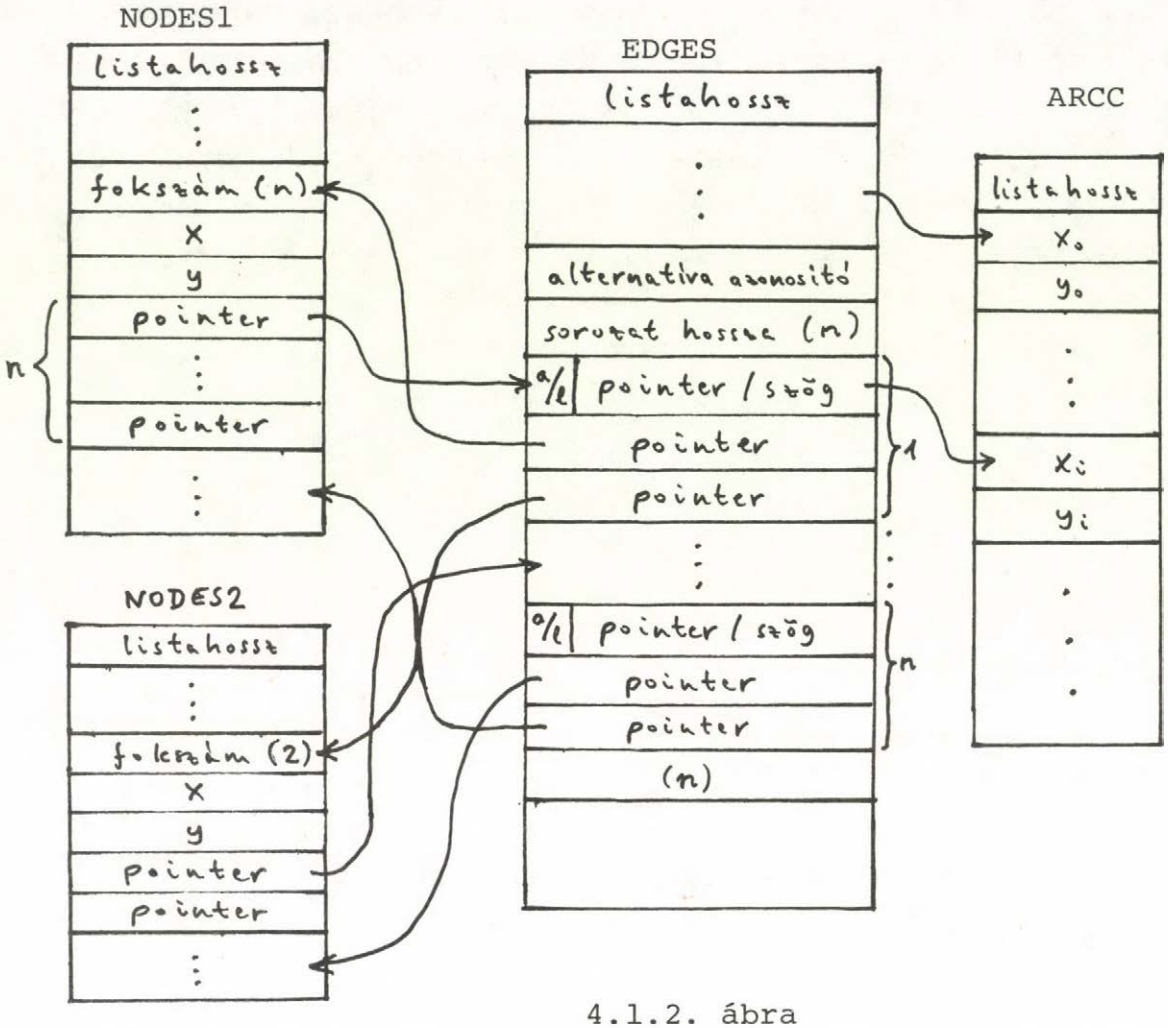
Az algoritmusok outputját négy /NODES1, NODES2, EDGES és ARCC/ lista formájában kapjuk meg. A NODES1 lista tartalmazza a látványgráf összes legalább harmadfoku csomópontját. A NODES2 lista a látványgráf másodfoku csucsainak töréspontjait tartalmazza. /Ha a 3.4. pontban leirt algoritmus után esetleg egyazon élsorozat több értelmezésében is szerepel ugyanaz a töréspont, akkor ezt is értelemszerüen mindig ujra felvesszük a NODES2-listába./ E két listában a csucsok sorban egymás után állnak, mindegyik a következő jellemzőkkel: x-koordináta, y-koordináta, fokszám és fokszámnak megfelelő számu pointer a csucsban csatlakozó élekre /4.1.2. ábra/

A látványgráf élei, illetve az egymás alternativáiként szereplő vonalsorozatok az EDGES-listában vannak egymás után leirva. Mindegyik vonalsorozat, ha n élből áll, 3n+3 szóban van leirva az EDGES-listában. Az első szóban egy számot helyezünk el, amely azt hivatott jelezni, hogy a szóbanforgó vonalsorozat melyik másik vonalsorozatnak alternativája. /Az alternativ sorozatokhoz azonos szám tartozik./ A második szóban azt irjuk le, hogy ez a vonalsorozat hány egyenes ill. körivdarabból áll. Ezután mindegyik vonalat 3 szóban irunk le, az első szó első bitje azt mondja meg, hogy ez a vonal egyenes vagy köriv, a szó további része pedig egyenes esetén az iránysz̈öget, köriv esetén pedig egy az ARCC listába mutató pointert tartalmaz.

Az ARCC listában sorra minden köriv egy-egy közbülső pontjának x- ill. y-koordinátáját irjuk be. Az egyes vonalak második és harmadik szava a vonal két végpontjára mutató pointert tartalmaz, ezek a NODES1 ill. NODES2 listába mutatnak. Végül a vonalsorozat utolsó szavába ismét felirjuk az illető vonalsorozat hosszát, ez a lista bejárását lényegesen megkönnyiti, ha a csomópontokból kell kiindulni.

Végül mind a négy lista elejére odairjuk a megfelelő lista hosszát. Ebből a listastrukturából a látványgráf minden tulajdonsága könnyen kiolvasható.

Az 1.1. pontban leirt algoritmust ugy implementáltam, hogy egy adott ablakméretre program generálja azt a programot, amely



4.1.2. ábra

magát az élkeresést végzi. Ez a programgeneráló program a feldolgozás elején néhány ezredmásodperc alatt lefut, és eredményeként az élkeresését egy, az adott ablakméretre specializált /s igy maximálisan gyors/ program végzi.

A többi /1.2., 2.2., 3.1., 3.3. és 3.4./ algoritmust a leirás szerint implementáltam egy 24 k szó memóriáju R10 gépen, a kvázi--párhuzamos algoritmusokat természetesen n = 1 processzor esetére.

Az input kép és az eredmények kijelzését egy Tektronix 613-as 256 x 256 képpont felbontásu tárolócsöves displayen oldottuk meg. Báthor Miklós [17] készitett egy programcsomagot, amelynek segitségével tetszőleges pontokból, egyenesekből és körivekből álló

ábrát ki lehet rajzoltatni a displayre. Az ábra bármelyik részén tetszőleges lineáris transzformációt végre lehet hajtani.

4.2. Kisérletek zajos generált képekkel

A lézer képpel dolgozó algoritmusokat, egyenlőre, müködő input eszköz hiján, szimulált zajos képekre próbáltuk ki. Ebből érdekes eredményeket kaptunk a soros eljárás zajérzékenységére. Az input képet a következőképpen szimuláltuk: Először rajzoltunk egy hibátlan rajzot az emlitett programcsomag segitségével. A rajz minden pontját p_o valószinüséggel kitöröltük az ábrából. Ezután minden pontból k pontnyi távolságra lefelé, felfelé, jobbra és balra p_k valószinüséggel ujabb /zajos/ képpontokat generáltunk. /k = 1, 2, 3, 4-re/. Az ábrából kitörölt pontok körül p_1 helyett p_1' valószinüséggel vettünk fel uj pontokat, mivel a lézer képben szakadás nemigen fordul elő. Ezután a display minden pontjában p_g valószinüséggel felvettünk egy zajpontot, tehát az egész ábrát "megsóztuk" zajjal. /Ilyenféle hibákat okoznak a kábelzajok./ A 4.2.1. ábra egy szintérnek az emlitett rajzoló programcsomag segitségével készitett képét mutatja. A 4.2.2. ábra a rajz $p_o = 0,3$ $p_1 = 0,25$ $p_1' = 0,4$ $p_2 = 0,2$ $p_3 = 0,05$ $p_4 = 0$ $p_g = 0,009$ értékek szerint megzajositott változata. A 4.2.3. ábra a 2.1. algoritmus 6. lépése előtt kapott eredményeket /az éldarabokat és a csomópont-gyanus helyeket/ mutatja. A 4.2.4. ábrán a 2.2. algoritmus végeredménye látható.

A 4.2.5. ábrán egy satu rajza látható, ezt az ábrát a display felbontóképességének növelése céljából két részre vágtuk. A baloldali részt /4.2.6. ábra/ $p_o = 0,2$ $p_1 = 0,2$ $p_1' = 0,33$ $p_2 = 0,08$ $p_3 = p_4 = 0$ $p_g = 0,006$
a jobboldali részt pedig /4.2.7. ábra/ $p_o = 0,15$ $p_1 = 0,25$ $p_1' = 0,4$ $p_2 = 0,25$ $p_3 = 0,08$ $p_4 = 0$ $p_g = 0,004$
értékek szerint zajositottuk. Az éldarabok és a csomópontjelöltek a 4.2.8. és 4.2.9. ábrán látható. A 4.2.10. ábra mutatja az algo-

ritmus végeredményét, ujra egyesitva a kép két felét.

Az ilymódon generált képekkel elvégzett kisérletek eredményeit ugy összegezhetjük, hogy elég jó látványgráfokat kaptunk, ha $p_3 = p_4 = 0$, $p_o < 0,1$ vagy tetszőleges p_o-ra, ha $p_1' > 0,33$. Ha $p_3 < 0,1$ vagy $p_3 > 0,2$ és $p_4 < 0,03$, akkor az eredmények erősen függtek attól, hogy hogyan választottuk meg az algoritmus paramétereit. Ezekben az esetekben meg lehetett választani a paramétereket ugy, hogy elfogadható látványgráfokat kapjunk. A bemutatott példák azokkal a paraméterekkel futottak, amiket az algoritmusok leirása során megemlitettünk. Érdekes, hogy az algoritmus müködése szinten egyáltalán nem volt érzékeny p_g-re /amig $p_g < 0,01$/.

Ezek a példák igen jellemzőek a soros algoritmus müködésére, látható, hogy hosszu és különálló vonalakat jól tud követni, de ahol a folytatás nem egyértelmü, ott gyakran utat téveszt, vagy hibás csomóponthoz köti az elkezdett utat /pl. a 4.2.10. ábra baloldalán, vagy a satu csavarjai körül/. A 2.2. algoritmus gyengéi is megmutatkoznak: pl. a 4.2.2. ábrán egy-egy hibás éldarab fölöslegesen megtörte az egybetartozó vonalakat. A kevéssé zajos részeken viszont jól követte az algoritmus a konturvonalakat.

Az algoritmus futási ideje a 4.2.1. ábrára 2,5 sec, a 4.2.5. ábrára 3,6 sec, a 4.2.6. ábrára pedig 4,8 sec volt.

4.3. Az uj élkereső algoritmus vizsgálata

Az 1.1. pontban már bemutattunk néhány példát az élkereső algoritmus müködésére.

A 4.3.1. ábrán egy, a szintérre helyezett félgömb látható, a 4.3.2. ábrán pedig ennek a TV-inputból kapott digitalizált képe. /A TV képernyője a digitalizált képet negativan jelzi ki./ A gömbfelület, mint a digitalizált kép is mutatja, azért érdekes választás, mert a fényesség folytonosan változik rajta, és ez a digitalizált képen szintvonalszerü fényességváltozásokban jelentkezik, és a változások határa meglehetősen zajos. A képet 3 pontnyira átfedő 6 x 6-os ablakokkal fedtük le, igy erre az egyetlen ábrára is

már az élkereső algoritmust kb. 3000-szer kellett végrehajtani. A 4.3.3. ábrán az algoritmus által talált élek láthatók. Látható, hogy az egyenesdarabok igen jól követik a fényességváltozások vonalát.

A tárgy felismerését azonban ezek a belső vonalak nagymértékben megnehezitik. Ezért az algoritmust ugy módositottuk, hogy csak akkor találjon élet, ha az ablakokban a szürkeségi szint változása legalább 2, azaz a 1.1. pont jelölésével $d \geq 2$. Az igy kapott éleket mutatja a 4.3.4. ábra. Algoritmusunknak ez a módositása azt a feltételezést foglalja magában, hogy a különböző lapok találkozásánál a szürkeségi szint legalább 2-vel változik, mig az 1 értékü változásokat a digitalizálás esetlegességének tekintjük. Ez a feltételezés itt is, és a további kisérletek során is helyesnek bizonyult.

A 4.3.5. ábrán ugyanezt a szinteret láthatjuk, de a digitalizálásnál minden második szürkeségi szintet kikapcsoltunk, tehát minden árnyalatkülönbség 2 lépcsős ugrást jelent. Az igy kapott képben talált élek láthatók a 4.3.6. ábrán.

Egy teljes képen az összes éldarabok megkeresése 6-8 másodpercet vesz igénybe, természetesen ez az idő 1 processzorral való feldolgozásra vonatkozik.

4.4. TV-képek feldolgozása

Ebben a pontban a 3. fejezetben leirt algoritmusokkal kapott eredményeket tekintjük át. A teljes feldolgozó algoritmust a következő öt szempont szerint vizsgáljuk:

a/ mennyire érzékeny a kép digitalizálásának esetlegességeire, hibáira és zajaira;

b/ mennyire érzékeny a megvilágitás szingularitásaira;

c/ adott képfelbontás mellett mennyire részletgazdag szintér feldolgozására képes;

d/ átlagos bonyolultságu szintér és jó megvilágitás esetén mennyire megbizható;

e/ futási idő.

Egy átlagos bonyolultságu szinteret ábrázol a 4.4.1. ábra. A 4.4.2 ábrán ennek a digitalizált képe látható /ismét negativ kép/. A 4.4.3. ábrán az élkereső eljárás átlal talált élek láthatók, a 4.4.4. ábra pedig a 3.1. pontban leirt eljárással kapott csomókat és a 3.3. pont algoritmusával kapott optimális élsorozatokat mutatja. A 4.4.5. ábrán a 3.4. pont algoritmusával kapott vonal-alternativák láthatók. A 4.4.6. ábrán ezek közül külön kirajzoltuk azokat, amelyek a felismerésben szerepet játszanak. Ezek egy kivétellel egyben azok a vonalalternativák is, amelyekhez a legnagyobb valószinüség-értékek tartoznak. /A kup alaplapja baloldalánál a köriv-értelmezéshez rendelt érték 0,15, mig az egyenes-értelmezéshez 0,85/.

Ez a példa az algoritmus müködésére jellegzetesnek mondható: hasonló bonyolultságu és megvilágitásu szinterekre az algoritmus az esetek több, mint 80%-ában hasonlóan jó eredményt ad. Ha a szintéren csak egy ilyenfajta tárgy van /azaz arra a felbontás nagyobb, mint az itteniekre/, akkor az esetek több, mint 95%-ában kapunk ilyen jó eredményt. Ez azt mutatja, hogy az a/ szempontban felvetett kérdésekre a válasz pozitiv: a feldolgozást az adott TV input eszköz zajai és digizalizálási esetlegességei /pl. a szintvonalszerü zajos szintváltások az egyes lapokon belül/ szinte egyáltalán nem zavarják.

A b/ szempontban felvetett probléma vizsgálatát a lámpák mozgatásával végeztük. Azt tapasztaltuk, hogy a megvilágitás kétféle problémát okozhat: az egyik az, ha két szomszédos lap szürkeségi szintje azonos, vagy különbségük csak 1; ebben az esetben a két lap határvonalát nem találjuk meg. Ez a probléma megfelelő megvilágitás esetén csak nagyon speciális esetekben merül fel, és természetesen az algoritmusokból nem is várhatjuk el, hogy két egyforma szürkeségü lap között élet találjon. /Ez esetben a későbbi, felismerő algoritmusoknak fog kelleni vagy hibát jelezni, vagy rájönni, hogy valahol egy él hiányzik./

A másik probléma az, ha egy görbe felületre ugy esik a fény, hogy a fényességváltozás gradiense valahol a felületek belsejében

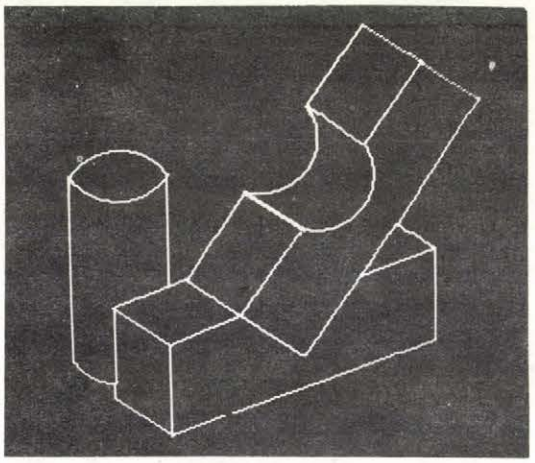
nagy, és ezért a "szintvonalak" a felületen belül valahol tul sürün vannak. /Ez az eset akkor fordul elő, ha az erős fényforrás iránya a tárgyra a kameráéval nagy szöget zár be./ Ilyenkor a sürü szintvonalak helyén az algoritmus fölösleges konturvonalakat talál. A 4.4.7. ábrán látható input képen jól tanulmányozható, hogy a szintvonalak milyen sürüsége az, ami még nem zavar, és mi az, ami már zavar. A szintéren a hátteret is meggörbitettük, hogy csillogjon. A 4.4.8-4.4.10. ábrák az egymásutáni algoritmusok eredményeit mutatják.

A 4.4.10. ábrán a baloldali test bevágásánál az algoritmus összekötött két valójában diszjunkt konturvonalat. Általában az algoritmus csak egymástól legalább 8-9 képpontnyi távolságra haladó konturvonalakt talál diszjunktnak, a közelebbieket összeköti, vagy, ha huzamosabban együtt haladnak, akkor csak az egyiket találja meg. Hasonló a helyzet a csomópontokkal is. A c/ szempontban felvetett kérdés vizsgálatához egy további sokatmondó szintér látható a 4.4.11. ábrán. /A további ábrákon sorra a digitalizált kép, az élek, a csomók és az optimális élsorozatok, végül a vonalalternativák láthatók./ Ennek a tárgynak a bonyolultsága a 144 x 192-es képfelbontás mellett meghaladja az algoritmus teljesitőképességét, de a vonalak többségét ennek ellenére megkapjuk. Ez a példa jól illusztrálja az algoritmusnak az emlitett tulajdonságait a megvilágitás szingularitásaival és a szintér részletgazdagságával kapcsolatban.

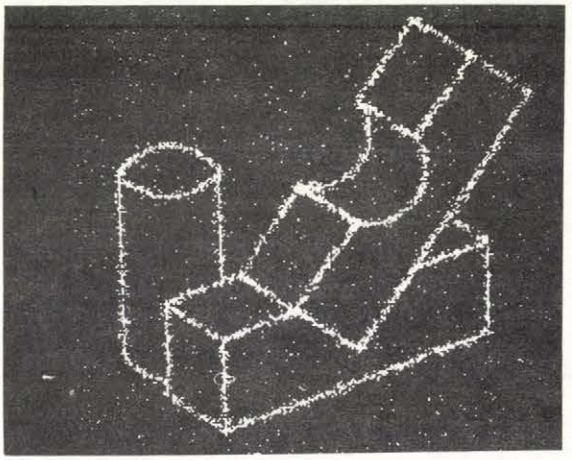
A konturvonalak meghatározása a digitalizált input képből összesen általában 15-25 másodpercet vesz igénybe. /A 4.4.2. ábrán látható kép feldolgozása 17, a 4.4.12. ábráé pedigg 22 sec volt./ Ebből az élkeresés 6-8 sec, a csomópontkeresés 2-3 sec, az optimális élsorozatok keresése 7-12 sec, végül a kontruvonal-alternativák megkeresésének ideje 1-4 sec. Ezek az időadatok természetesen egy /R10-es/ processzorral értendők. Külön lemértük a feltétlenül egymásután végzendő részalgoritmusok időszükségletét is, ez összesen 0,8 sec volt. Ez tehát azt jelenti, hogy ha a processzorok száma korlátlan, akkor ennyire szoritható le a feldolgozás ideje. Próbaként kiszámoltuk, hogy már 10 processzor is a végrehajtási időt kb. 2-2,5 sec-re csökkentené. Ez indokolja azt, hogy az algoritmu-

sok tervezésénél az is szempont volt, hogy azok kvázi-párhuzamosan is végezhetők legyenek.

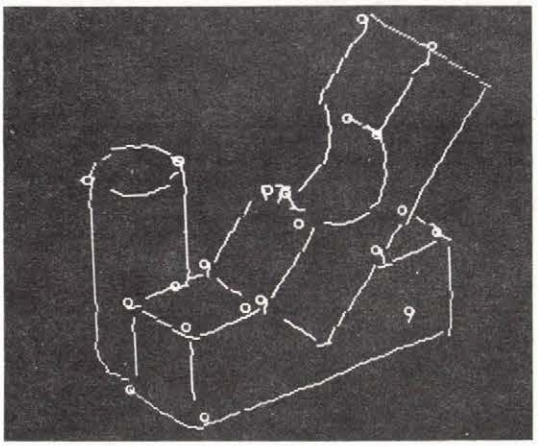
A kapott konturvonalakat az intelligens szem-kéz rendszerben ugy használjuk, hogy segitségükkel azonositjuk a szintéren lévő tárgyakra jellemző, egy előre elkészitett modellben meghatározott strukturákat. Igy ismerjük fel a tárgyakat és azonositjuk tényleges poziciójukat. Igy a látványgráftól a következő feldolgozási és manipulációs lépések azt követelik meg, hogy a felismeréshez szükséges vonalstrukturákat tartalmazza, és olyan pontossággal, hogy a kéz a tárgyat közre tudja venni és meg tudja fogni. A tárgy pontos helyzetét már a megfogás után a kéz ismert helyzete alapján határozzuk meg.



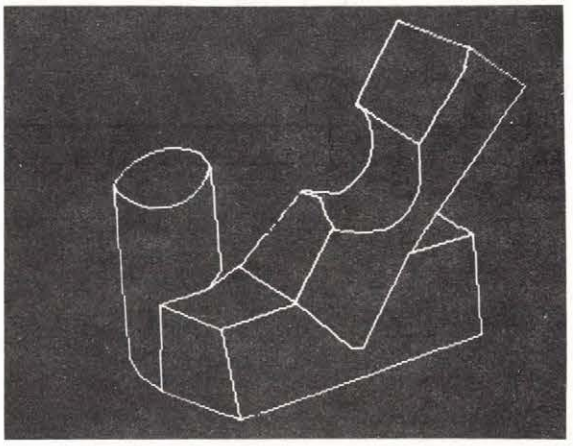
4.2.1. ábra



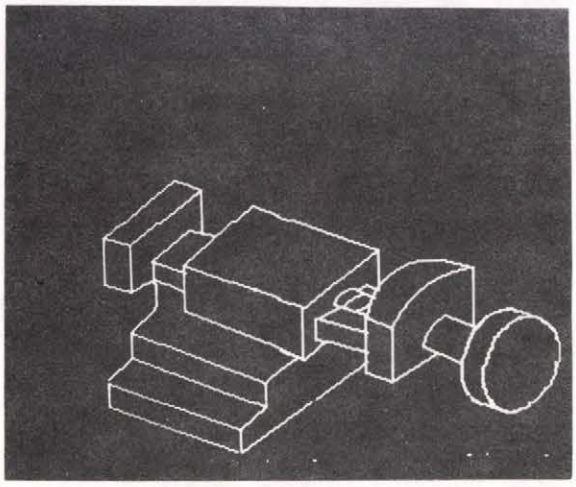
4.2.2. ábra



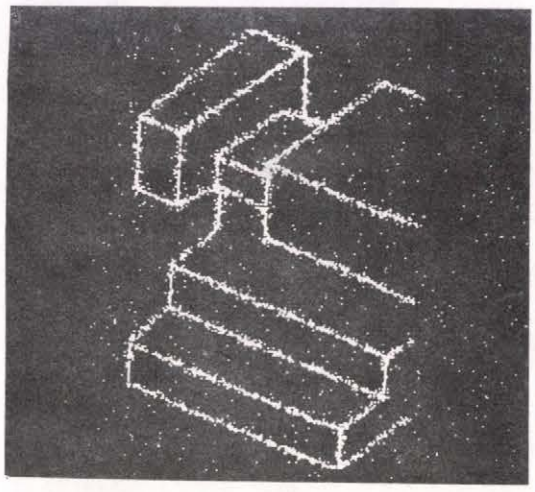
4.2.3. ábra



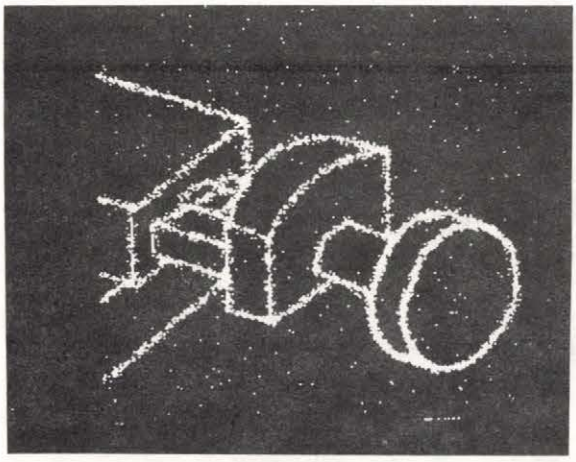
4.2.4. ábra



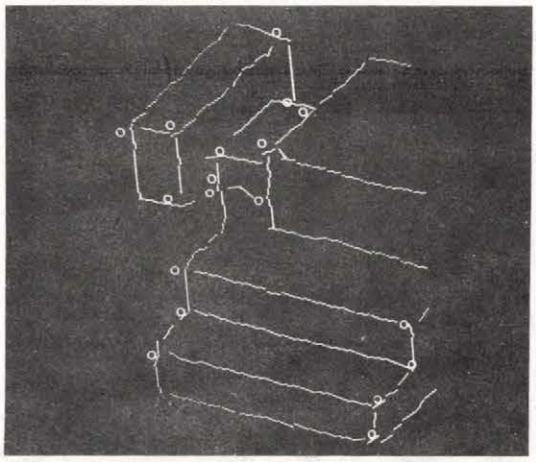
4.2.5. ábra



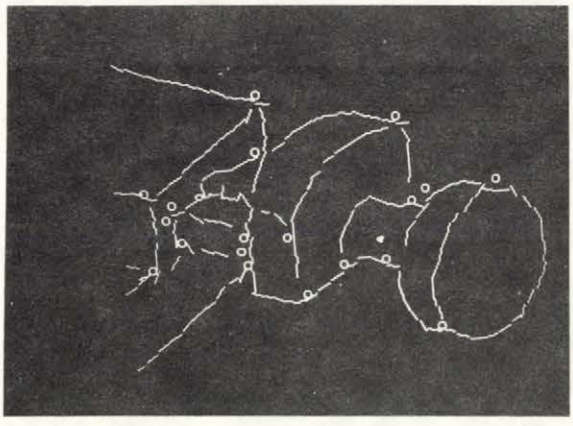
4.2.6. ábra



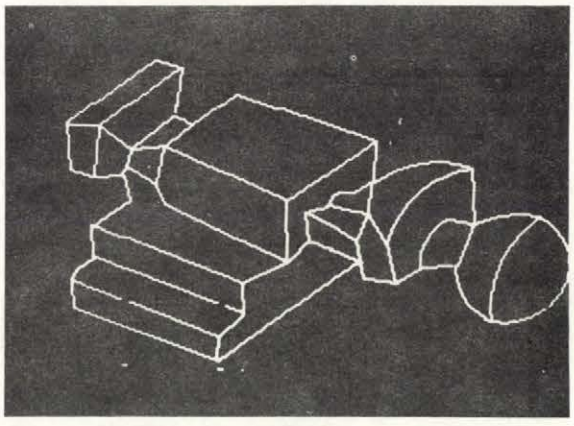
4.2.7. ábra



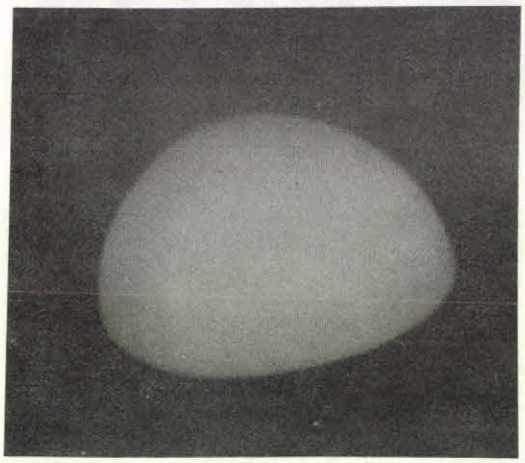
4.2.8. ábra



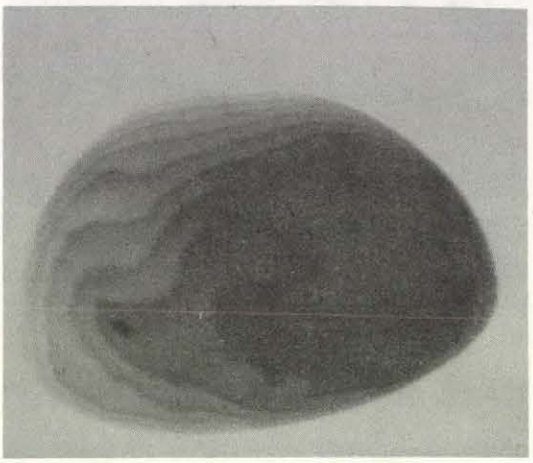
4.2.9. ábra



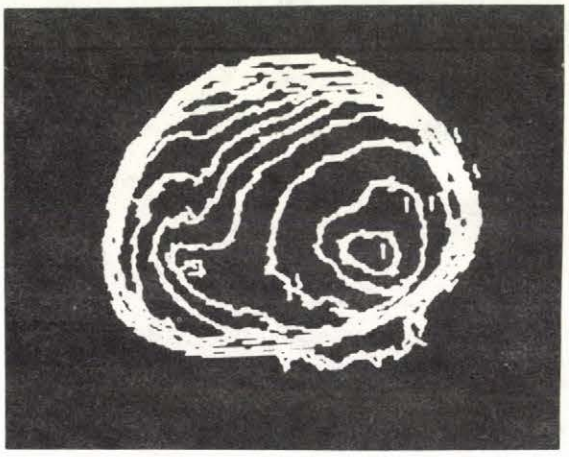
4.2.10. ábra



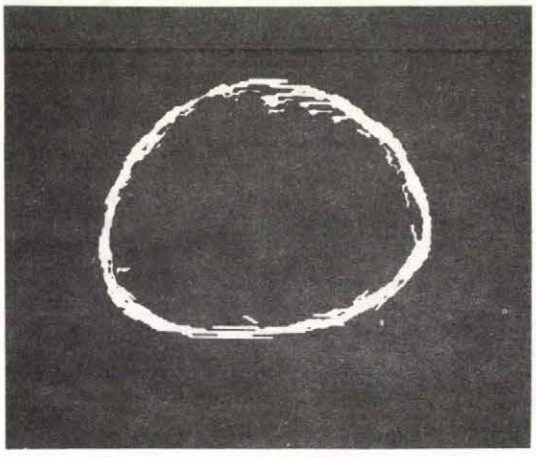
4.3.1. ábra



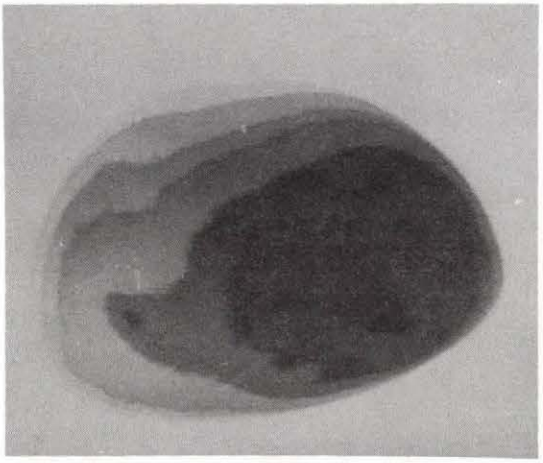
4.3.2. ábra



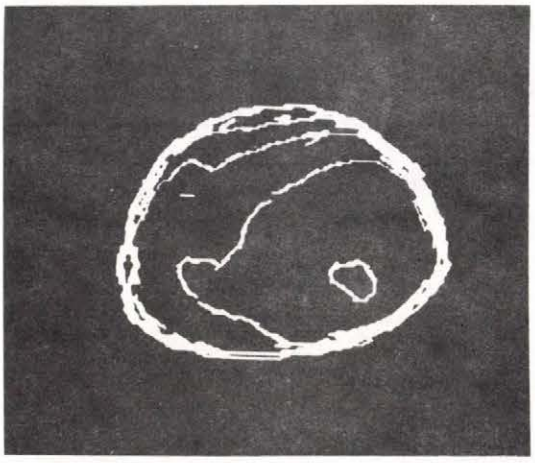
4.3.3. ábra



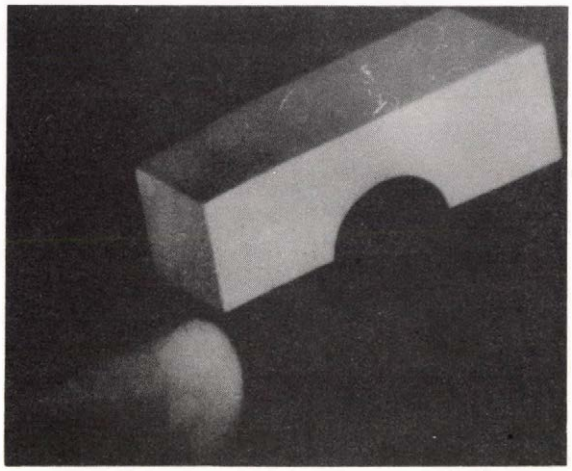
4.3.4. ábra



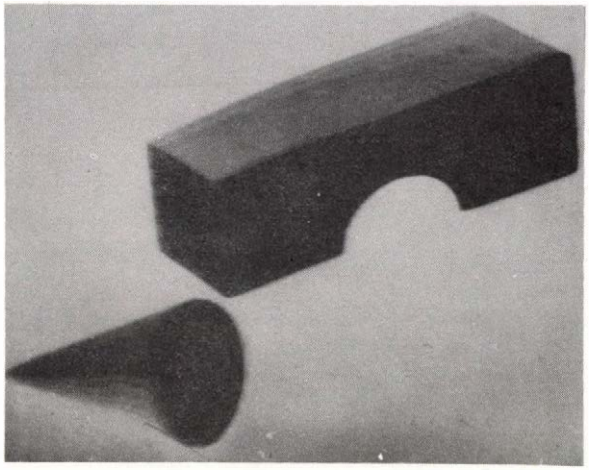
4.3.5. ábra



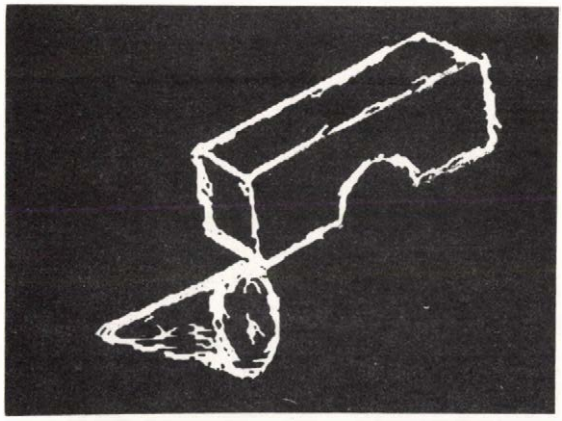
4.3.6. ábra



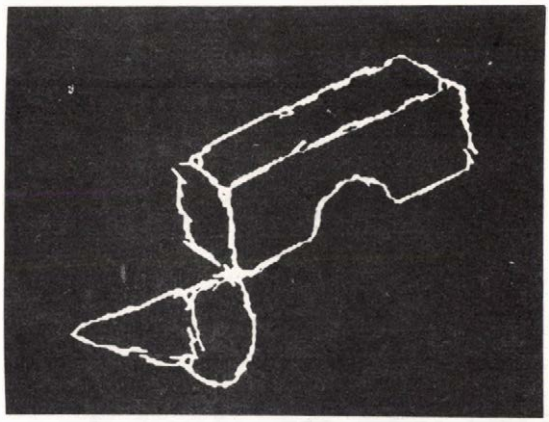
4.4.1. ábra



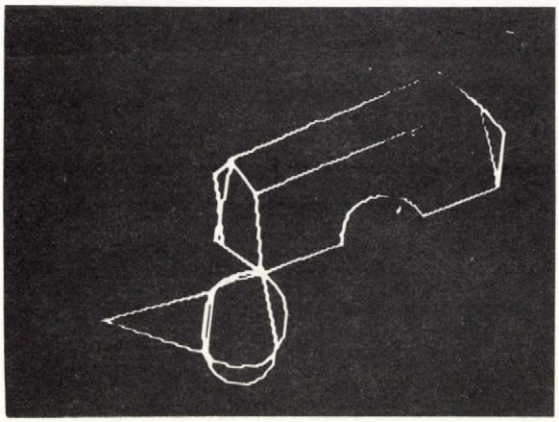
4.4.2. ábra



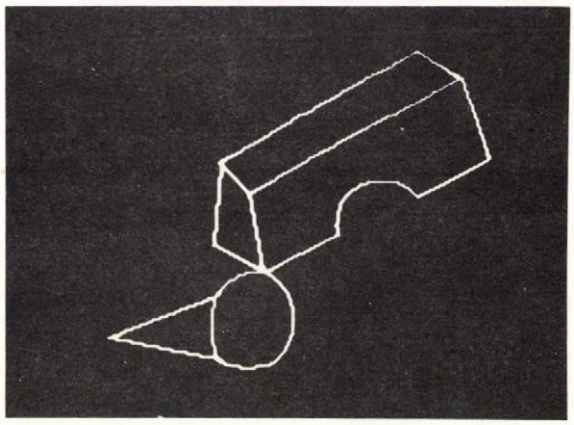
4.4.3. ábra



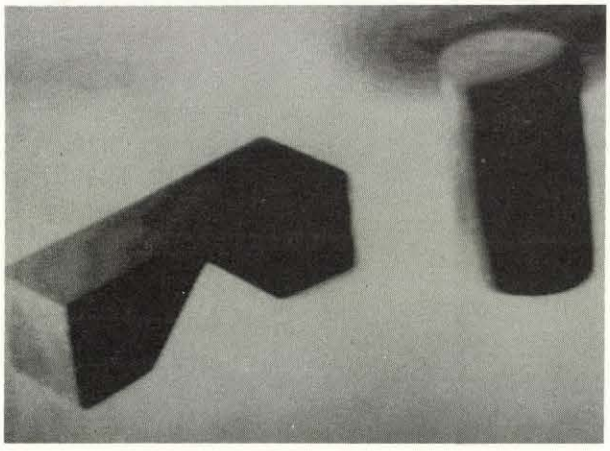
4.4.4. ábra



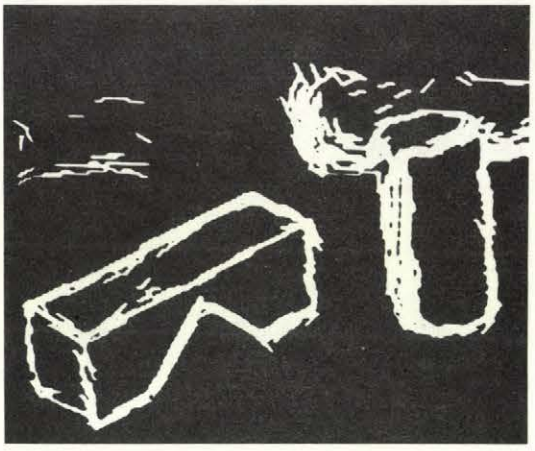
4.4.5. ábra



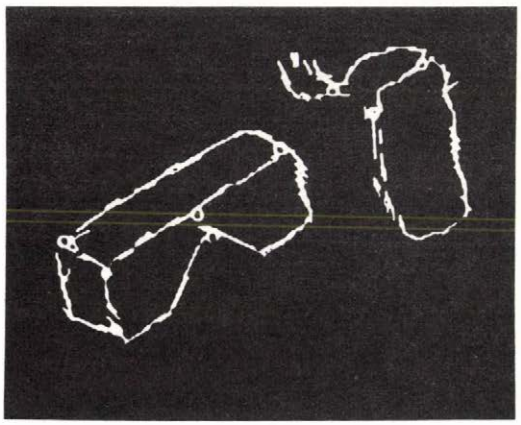
4.4.6. ábra



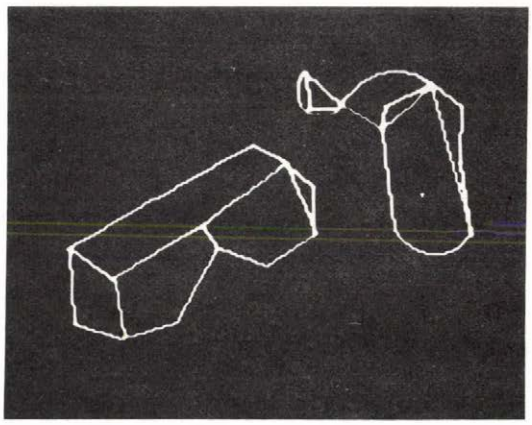
4.4.7. ábra



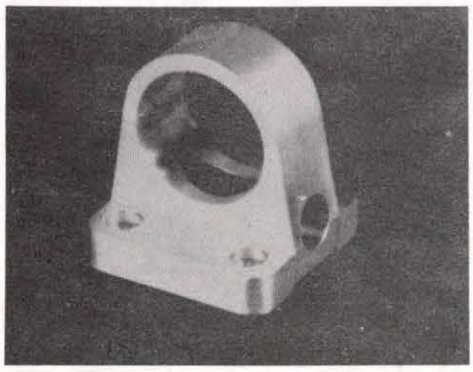
4.4.8. ábra



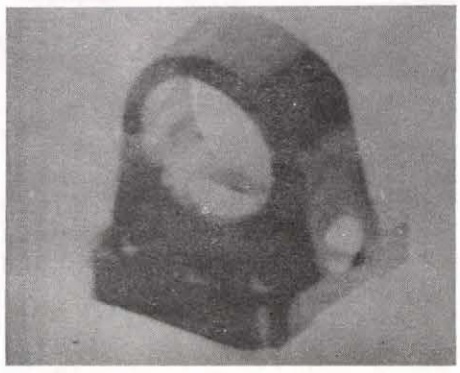
4.4.9. ábra



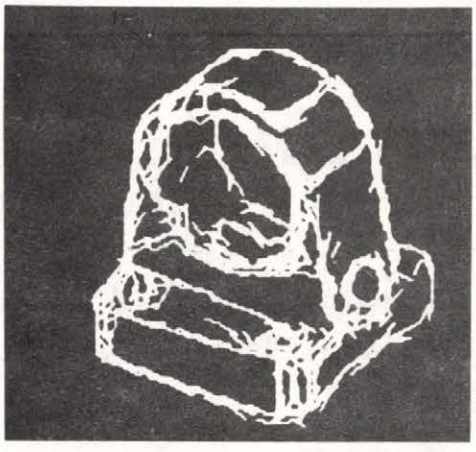
4.4.10. ábra



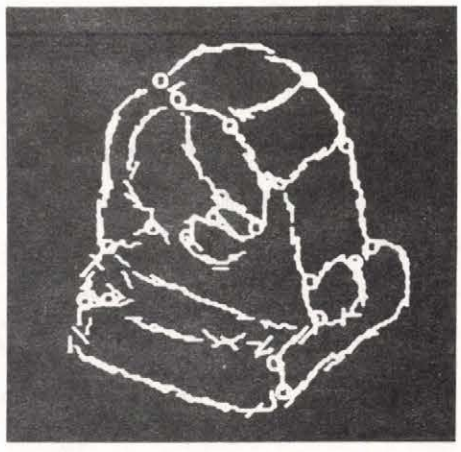
4.4.11. ábra



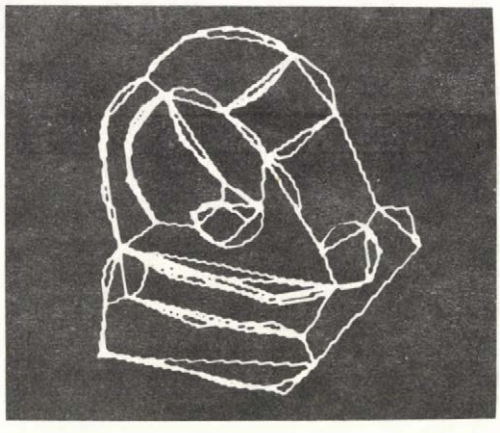
4.4.12. ábra



4.4.13. ábra



4.4.14. ábra



4.4.15. ábra

FÜGGELÉK

A Hueckel operátor

Hueckel [36] az optimális éldarab keresésének problémáját a következőképpen fogalmazza: Keressük meg az adott A ablakban a következő alaku foggvények közül azt az F(x,y) függvényt, amelyiknek az l_2 norma szerinti eltérése az f(x,y) input függvény A-beli megszoritásától minimális. F(x,y) alakja a következő legyen:

$$F(x,y,c,s,\rho,d,b) = \begin{cases} b, & \text{ha} \quad cx + sy \leqslant \rho \\ b+d, & \text{ha} \quad cx + sy > \rho \end{cases}$$

Az F(x,y) függvényt tehát egy ötparaméteres függvényseregből szeretnénk kiválasztani, ahol az öt paraméter (c,s,ρ,b,d) jelentése a következő: F(x,y) legyen olyan, hogy A-ban a $cx + sy = \rho$ egyenes fölött b, alatta pedig egy másik, b+d szürkeségi értéket vegyen fel, azaz F(x,y) legyen egy idealizált élfüggvény. A feladat tehát c,s,ρ,b és d meghatározása ugy, hogy

$$\delta(c,s,\rho,b,d) = \int_A \left(f(x,y) - F(x,y,c,s,\rho,b,d)\right)^2 dx\, dy$$

minimális legyen.

A c,s,ρ,b,d paraméterek megválasztásának utja a következő lesz: legyen $\{K_i\}^{\infty}$ az A-n értelmezett függvények Hilbert-terének egy /később specifikálandó/ ortonormált bázisa.

Legyen

$$a_i = \int_A K_i(x,y)\, f(x,y)\, dx\, dy$$

és

$$f_i(c,s,\rho,b,d) = \int_A K_i(x,y)\, F(x,y,c,s,\rho,b,d)\, dx\, dy$$

ekkor azt kapjuk, hogy

$$\delta(c,s,\rho,b,d) = \sum_{i=0}^{\infty} (a_i - f_i(c,s,\rho,b,d))^2$$

$\delta(c,s,\rho,b,d)$ ez utóbbi alakjának minimalizálásához néhány közelitő lépést kényszerülünk tenni.

Ahhoz, hogy az egyenesek helyét és iránytangensét kellemesen lehessen kezelni, célszerünek bizonyult a polárkoordinátás Fourier -analizis bázisfüggvényeit választani, Ez az oka annak, hogy az értelmezési tartományt kőrnek választjuk. A számitástechnikai végrehajthatóság érdekében a bázisnak csak az első 8 tagját fogjuk számolni, igy az eddigi szummákban a ∞ jel 7-tel helyettesithető. Legyen:

$$r = x^2 + y^2 , \qquad Q(r) = (1-r^2)^{1/2}$$

$$H_0(x,y) = \left(\frac{5}{6\Pi}\right)^{1/2} Q(r)\,(1 + 2r^2)$$

$$H_1(x,y) = \left(\frac{8}{27}\right)^{1/2} Q(r)\,(5r^2 - 2)$$

$$H_2(x,y) = \left(\frac{3}{\Pi}\right)^{1/2} Q(r)\,(4r^2 - 1)x$$

$$H_3(x,y) = \left(\frac{3}{\Pi}\right)^{1/2} Q(r)\,(4r^2 - 1)y$$

$$H_4(x,y) = \left(\frac{3}{\Pi}\right)^{1/2} Q(r)\,(3 - 4r^2)x$$

$$H_5(x,y) = \left(\frac{3}{\Pi}\right)^{1/2} Q(r)\,(3 - 4r^2)y$$

$$H_6(x,y) = \left(\frac{32}{3}\right)^{1/2} Q(r)(x^2 - y^2)$$

$$H_7(x,y) = \left(\frac{32}{3}\right)^{1/2} Q(r)\,2xy$$

$$K_0(x,y) = \left(\frac{24\Pi}{25}\right)^{1/2} H_0(x,y)$$

$$K_1(x,y) = \left(\frac{9}{2}\right)^{1/2} H_1(x,y)$$

$$K_2(x,y) = H_2(x,y) + H_4(x,y)$$

$$K_3(x,y) = H_3(x,y) + H_5(x,y)$$

$$K_4(x,y) = \frac{3}{2} H_6(x,y)$$

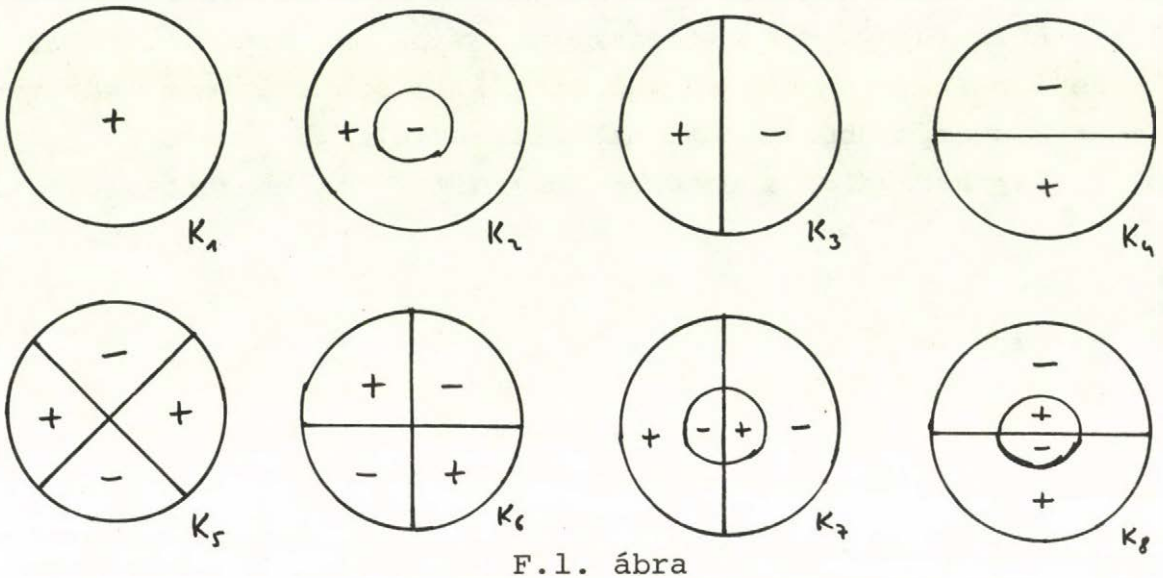
$$K_5(x,y) = \frac{3}{2} H_7(x,y)$$

$$K_6(x,y) = \left(\frac{5}{4}\right)^{1/2} H_2(x,y) - H_4(x,y)$$

$$K_7(x,y) = \left(\frac{5}{4}\right)^{1/2} H_3(x,y) - H_5(x,y)$$

$\{K_i(x,y)\}_{i=1}^{7}$ végül is a Hilbert tér igért ortonormált bázisa. A $\{H_i(x,y)\}_{i=1}^{7}$ függvények alkalmazása sok számitást takarit meg. A

bázisfüggvények viselkedését az F.1. ábra mutatja: a körök határvonalán, és a belül jelzett vonalakon a bázisfüggvények értéke 0, másutt a jelzett előjelüek.



F.1. ábra

$\delta(c,s,\rho,b,d)$ minimalizálásának végrehajtása egy elegáns tételen alapul, amelynek kimondásához még néhány mennyiséget kell definiálnunk: legyen

$$e_0(c,s) = a_2c + a_3s$$
$$e_1(c,s) = a_4c + a_5s$$
$$e_2(c,s) = a_1 + a_6(c^2 - s^2) + 2a_7cs$$
$$U(c,s) = \frac{e_0(c,s)}{e_0(c,s)} \left(e_1^2(c,s) + e_2^2(c,s)\right)^{1/2}$$
$$A(c,s) = e_0(c,s) + U(c,s)$$

Most már megfogalmazhatjuk a megoldó tételt:

Tétel: $\delta(c,s,\rho,b,d)$ globális és lokális minimumainak helyei egybeesnek $A(c,s)$ globális és lokális szélsőértékeivel.
Ezeken a helyeken:

$$\rho = \frac{e(c,s)}{(U(c,s) + e(c,s))\sqrt{2}}$$

$$d = \frac{4\,A(c,s)}{(1-\rho^2)^2(1+2\rho^2)\sqrt{3\Pi}}$$

$$b = A(c,s)\,\frac{(4+\rho(3+\rho\ (2+\rho)))d\ (1-\rho)^2}{8}$$

A tétel bizonyitása megtalálható Hueckel idézett dolgozatában

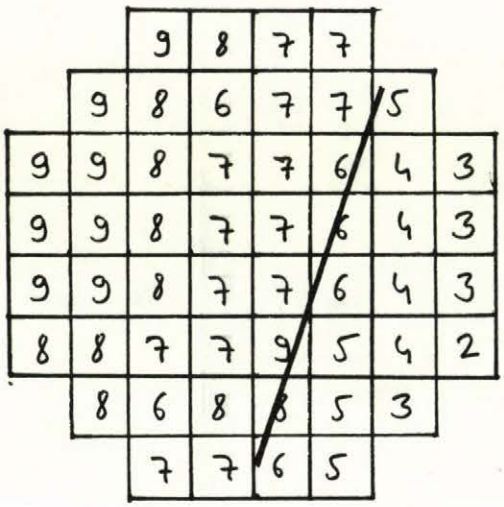
A(c,s) szélsőértékei már nehézség nélkül meghatározhatók, igy a keresett optimális élet megkaptuk. Több szélsőérték esetén valószinüleg több egyenes él is van a képen, de ebben az esetben, mint Hueckel megjegyzi, ha csak egy él markáns, azt még viszonylag pontosan megkapjuk, különben az eredmény nem nagyon megbizható.

Szellemes, és az eddigi részeredményekből jól számitható módszerrel méri Hueckel a talált él "jóságát": az f és a talált F függvény Hilbert térbeli szögének socinusával:

$$k = \frac{f \cdot F}{\|f\|\|F\|} = \left| \frac{A(c,s)}{(6a_1^2 + 2(a_2^2 + a_3^2 + a_4^2 + a_5^2) + 3(a_6^2 + a_7^2))^{1/2}} \right|$$

Ha f már maga is idealizált él, akkor k = 1. Az operátor a talált élet tul zajosnak nyilvánitotta k < 0,9 esetén.

Az 1.1.2. ábra egy példa az operátor müködésére, az eredmény itt c = - 0,99, s = 0,14, ρ = -0,28, b = 4,08, d = 3,80 k = 0,95 A képbe bejelöltük az igy kapott él helyét.


9 8 7 7
9 8 6 7 7 5
9 9 8 7 7 6 4 3
9 9 8 7 7 6 4 3
9 9 8 7 7 6 4 3
8 8 7 7 9 5 4 2
8 6 8 8 5 3
7 7 6 5

F.2. ábra

IRODALOM

Általános müvek

[1] I.E.Abdou: Quantitative Methods of Edge Detection, Univ. of Southern California, USCIPI Report No. 830.

[2] H.C.Andrews: Computer Technics in Image Processing, Academic Press, New York, 1970.

[3] L.S.Davis: A Survey of Edge Detection Technics, Computer Graphics and Image Processing 4 /1975/, pp 248-270.

[4] R.O.Duda-P.E.Hart: Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[5] J.R.Fram-E.S.Deutsch: On the Quantitative Evaluation of Edge Detection Shemes and Their Comparison with Human Performance, IEEE Trans. on Comp., Vol. C-24, No 6, /1975/, pp 616-628.

[6] Fritz,J.-Révész,P.: Az alakfelismerés statisztikus módszerei, Budapest, 1974.

[7] K.S.Fu: Syntactic Methods in Pattern Recognition, Academic Press, New York, 1974.

[8] U.Grenander: Pattern Synthesis, New York-Heidelberg-Berlin, Springer, 1976.

[9] H.J.Nilsson: Problem Solving Methods in Artifical Intelligence, New York, McGrew Hill, 1971.

[10] A.Rosenfeld: Picture Processing by Computer, Academic Press, New York, London, 1969.

[11] A.Rosenfeld: Progress in Picture Processing: 1969-1971, Comp. Survey, 5, /1973/, pp 81-108.

[12] A.Rosenfeld-A.C.Kak: Digital Picture Processing, Academic Press, New York-San Francisco-London, 1976.

[13] T.Vámos: Industrial Objects and Machine Part Recognition, Applications on Syntactic Pattern Recognition, Chapter 10 /ed.: K.S.Fu/, Springer, Heidelberg, 1977.

[14] T.Vámos-Z.Vassy: The Budapest Robot - Pragmatic Intelligence, Proc. of the 6th World Congress of IFAC, Boston, USA, 1975.

[15] T.Vámos-Z.Vassy: Industrial Pattern Recognition Experiment - a Syntax Aided Approach, Proc. 1. IJCPR, Washington, 1973, pp 445-452.

[16] T.Vámos-M.Báthor-L.Mérő: A Knowledge-Based Robot Vision System, megjelenés alatt

[17] P.H.Winston /ed./: The Psychology of Computer Vision, McGrew Hill, New York, 1975.

A dolgozatban idézett cikkek

[18] A.P.Ambler-H.G.Barrow-C.M.Brown-R.M.Burstall, R.J.Poppelstone: A Versatile Computer-Controlled Assembly System, Proc.3. DCAI /Stanford/, 1973, pp. 298--307.

[19] M.Báthor: Interactive Picture Manipulation, Preprints of the Second Hungarian Computer Science Conf., Budapest 1977, pp 168-177.

[20] M.J.Brooks: Rationalizing Edge Detectors, Comp. Graph. and Im.Proc. 8 /1978/, pp 277-285.

[21] C.K.Chow-T.Kanako: Automatic Boundrary Detection of the Left Ventricle from Cineangiograms, Comp.Biomed.Res. 5 /1972/, pp 388-410.

[22] L.Cordella-M.J.Duff-S.Levialdi: Comparing Sequential and Parallel Processing of Pictures, 3 IJCPR, Coronado, California, 1976, pp 703-707.

[23] R.O.Duda- P.E.Hart: Use of the Hough Transformation to Detect Lines and Curves in Pictures, Comm.ACM, 11 /1972/, pp 11-15.

[24] R.O.Duda-D.Nitzan: Low-Level Processing of Registered Intensity and Range Data, 3 IJCPR, Coronado, California, 1976, pp 598-601.

[25] H.Freeman-L.S.Davis: A Corner-Finding Algorithm for Chain--Coded Curves, IEEE Trans.Comp.C-26 /1977/ pp 297-303.

[26] W.Frei-Chung-Ching Chen: Fast Boundary Detection at Generalization and a New Algorithm, IEEE Trans.Comp.C-26, /1977/, pp 988-998.

[27] V.Galló: A Program for Grammatical Pattern Recognition Based on the Linguistic Method of the Description and Analysis of Geometrical Structures, 4th IJCAI, Tbilisi, USSR, 1975, pp 628-634.

[28] V.Galló: Szisztyema dlja abrabotki szpiszkov dlja intelligentnovo robota /oroszul/ 2. Hungarian Comp.Sci.Conf., Budapest, 1977, pp. 400-411.

[29] A.K.Griffith: Edge Detection in Simple Scenes Using a priori Information, IEEE Trans. on Comp. C-22 /1973/, pp 371-381.

[30] A.K.Griffith: Mathematical Models for Automatic Line Detection, J.ACM., 20 /1973/, pp 62-80.

[31] M.Hajnal-I.Loványi-L.Mérő-A.Siegler-L.Vajta: Egy számitógéppel vezérelt képfeldolgozó berendezés és felhasználása egy intelligens szem-kéz rendszerben, Mérés és Automatika, 1978, pp 255-258.

[32] R.M.Haralick-J.S.Kartus: Arrangements, Homomorphismus and Discrete Relaxation, IEEE Trans. SMC, 8 /1978/, pp 600-612.

[33] G.T.Herman-A.Lent-P.H.Lutz: Relaxation Methods for Image Reconstruction, Comm.ACM, 21 /1978/, pp 152-158.

[34] F.Holdermann-H.Kazmierczak: Processing of Gray-scale Pictures, Comp.Graph. a Im. Proc. 1 /1972/, pp 66-80.

[35] D.H.Hubel-T.N.Wiesel: Receptive Fields, Binocular Interaction, and Functional Architecture of the Cat's Visual Cortex, J. Physiol.160 /1962/, pp 106-154.

[36] M.H.Hueckel: An Operator which Locates Edges in Digitized Pictures, J.ACM.18 /1971/, pp 113-125.

[37] M.H.Hueckel: A Local Visual Operator which Recognizes Edges and Lines, J.ACM.20 /1973/, pp 634-647.

[38] M.Ishii-T.Nagata: Feature Extraction of Three-dimensional Objects and Visual Processing in a Hand-Eye System Using Laser Tracker, Pattern Recognition 8 /1976/, pp 229-237.

[39] N.F.Izzo-W.Coles: Blood Cell Scanner Identifies Rare Cells, Electronics 35, /1962/, pp 52-57.

[40] T.Kasvand: Iterative Edge Detection, Comp.Graph. a Image Proc. 4 /1975/, pp 279-286.

[41] B.Kruse: A Parallel Picture Processing Machine, IEEE Trans., Comp.C-22 /1973/, pp 1075-1087.

[42] S.Levialdi: On Shrinking of Binary Patterns, Comm.ACM.15, /1972/, pp 7-10.

[43] A.Martelli: Edge Detection Using Heuristic Search Methods, Comp.Graph. and Image Proc. 1 /1972/, pp 169-182.

[44] L.Mérő: The Implementation of a Fast Contour-Searching Algorithm in TV or Laser Pictures on the R10 Minicomputer, 2nd Hung.Comp.Sci.Conf., Budapest, 1977, pp 663-672.

[45] L.Mérő: A Quasi-Parallel Contour Following Algorithm, Proc. /AISB/GI Conf.on AI., 1978,pp 189-194.

[46] L.Mérő-Z.Vassy: A Simplified and Fast Version of the Hueckel Operator for Finding Optimal Edges in Pictures, 4th IJCAI Tbilisi, USSR, 1975, pp 650-655.

[47] L.Mérő-T.Vámos: Real-time Edge Detection Using Local Operators, 3rd IJCPR Coronado, California, 1976, pp 31-36.

[48] L.Mérő-T.Vámos: Real-time Contour Detection, megjelenés alatt

[49] U.Montanari: On the Optimal Detection of Curves in Noisy Pictures, Comm.ACM., 14 /1971/ pp 335-345.

[50] J.P.Mylopoulos-T.Pavlidis: On the Topological Proporties of Quantized Spaces I-II. J.ACM. 18 /1971/, pp 239-254.

[51] N.E.Nahi-S.Lopez-Mora: Estimation-Detection of Object Boundaries in Noisy Images, IEEE Trans., 23 /1978/, pp 834-846.

[52] J.von Neumann: Theory of Self-Reproducing Automata, Urbana, Illinois, 1966.

[53] F.O'Gorman: Edge Detection Using Walsh Functions, Artificial Intelligence 9 /1978/, pp215-223.

[54] R.Nevatia: Evaluation of a Simplified Hueckel Edge-Line Detector, Comp.Graph. and Im.Proc. 6 /1977/, pp 582-588.

[55] W.A.Perkins-T.O.Binford: A Corner Finder for Visual Feedback, Comp.Graph. and Im.Proc. 2 /1973/, pp 355-376.

[56] E.Persoon: A New Edge Detection Algorithm and its Applications in Picture Processing, Comp.Graph. and Im.Proc.5, /1976/, pp 425-446.

[57] K.K.Pingle: Visual Perception by Computer, in "Automatic Interpretation and Classification of Images" /ed.: A.Grasselli/, Academic Press, New York, 1969, pp 277-284.

[58] E.U.Ramer: Transformation of Photographic Images into Stroke Arrays, IEEE Trans.Circ.Syst.CAS-22 /1975/, pp 363-373.

[59] N.S.Ramesh-K.S.Fu: A Survey of Computer Architectures for Image Processing and Pattern Recognition, Tech. Report TR-EE 77-38, Purdue Univ., Indiana, 1977.

[60] C.V.K.Rao-B.Prasada-K.R.Sarma: A Parallel Shrinking Algorithm for Binary Patterns, Comp.Graph. and Im.Proc.5 /1976/, pp 265-270.

[61] L.G.Roberts: Machine Perception of Three-Dimensional Solids, in Optical and Electrooptical Information Proc, MIT Press Cambridge, Mass., 1965, pp 159-197.

[62] A.Rosenfeld: Connectivity in Digital Pictures, J.ACM,17 /1970/, pp 146-160.

[63] A.Rosenfeld: A Characterization of Parallel Thinning Algorithms, Information and Control 29 /1975/, pp 286-291.

[64] A.Rosenfeld-L.S.Davis: A Note on Thinning, Tech.Report, R-809, Univ. of Maryland, 1975.

[65] A.Rosenfeld-R.A.Hummel-S.W.Zucker: Scene Labeling by Relaxation Operations, IEEE Trans.System, Man and Cybernetics SMC-6 /1976/, pp 420-433.

[66] A.Rosenfeld-R.Thomas-Y.Lee: Edge and Curve Enhancement in Digital Pictures, Univ.of Maryland Tech.Report 1969, pp 69-93.

[67] A.Rosenfeld-M.Thurston: Edge and Curve Detections for Visual Scene Analysis, IEEE Trans.Comp.C-20, /1971/, pp 562-569.

[68] A.Rosenfeld-M.Thurston-Y.Lee: Edge and Curve Detection: Further Experiments, IEEE Trans. Comp.C-21, /1972/, pp 677-715.

[69] B.J.Schachter-A.Lev-S.W.Zucker-A.Rosenfeld: An Application of Relaxation Methods to Edge Reinforcement, IEEE Trans.Systems, Man and Cyb., SMC-7, /1977/, pp 813-816.

[70] S.D.Shapiro: Aspects of Transform Method for Curve Detection, IEEE Publ. 76H1169-2 C, 1976.

[71] Shi-Kuo Chang: The Computation of Window Operations on a Parallel Organized Computer - A Case Study, IEEE Trans.Comp. C-22 /1973/, pp 34-40.

[72] Y.Shirai: Edge Finding, Segmentation of Edges and Recognition of Complex Objects, 4th IJCAI, Tbilisi, USSR, 1975, pp 674-682.

[73] Y.Shirai: Analyzing Intensity Arrays Using Knowledge about Scenes, [15] -ben.

[74] A.Siegler: Computer Controlled Object Manipulation, 2nd Hung. Com.Sci.Conf., Budapest, 1977, pp 724-738.

[75] J.Sklansky: Image Segmentation and Feature Extraction, IEEE Trans.SMC 8, /1978/, pp 237-247.

[76] I.Sobel: On Calibrating Controlled Cameras for Perceiving 3-D Scenes, Art.Int. 5 /1974/, pp 185-198.

[77] R.Stefanelli-A.Rosenfeld: Some Parallel Thinning Algorithms for Digitized Pictures, J.ACM, 18 /1971/, pp 255-264.

[78] T.Uno-M.Ejiri-T.Tokunaga: A Method of Real-Time Recognition of Moving Objects and its Application, Pat.Recog. 8 /1976/, pp 201-208.

[79] G.J.VanderBrug-A.Rosenfeld: Two-Stage Template Matching, IEEE Trans.Comp. C-26 /1977/, pp 384-393.

[80] D.Waltz: Understanding Line Drawings of Scenes with Shadows, [15] -ben.

[81] H.Wechsler-M.Kidode: A New Edge-Detection Technique and its Implementation, IEEE Trans. on Systems, Man and Cyb., SMC-7, /1977/, pp 827-836.

[82] J.Weszka-R.Nagel-A.Rosenfeld: A Technique for Facilitating Threshold Selection for Object Extraction from Digital Pictures, Univ. of Maryland, Tech.Report 243, 1973.

[83] M.Yachida-S.Tsuji: A Versatile Machine Vision System for Complex Industrial Parts, IEEE Trans.Comp. C-26, /1977/, pp 882-894.

[84] S.W.Zucker-R.A.Hummel-A.Rosenfeld: An Application of Relaxation Labeling to Line and Curve Enhancement, IEEE Trans.Comp. C-26 /1977/, pp 394-403.

[85] S.W.Zucker-E.V.Krishnamurty-R.L.Haar: Relaxation Processes for Scene Labeling: Convergence, Speed and Stability, Techn.Report 477, Univ.of Maryland, 1976.

További irodalom

Azriel Rosenfeld a Computer Graphics and Image Processing c. lapban évről-évre ir egy összefoglaló cikket, amelyben összefoglalja az adott év termését a képfeldolgozás területén, nagyon sok /rendszerint 3-400/ irodalmi hivatkozással. Ezeknek a cikkeknek az alapján a képfeldolgozás témakörének irodalma 1973. óta igen jól áttekinthető.

A T A N U L M Á N Y O K sorozatban 1978-ban megjelentek:

74/1978 Vortrüge über das graphische Display GD'71

75/1978 Vaskövi István - Galbavy Márta: Anyagszétválasztási rendszerek tervezésének és optimális üzemeltetésének általános megközelitése

76/1978 Somló János - Nagy Judit: Módszer munkadarabok forgácsoló megmunkálási folyamatának optimalizálására.

77/1978 Szászné Turchányi Piroska: Optimalizálási feladatok csomagkapcsolt számitógéphálózatok tervezésénél

78/1978 Darvas Péter - Gallai István - Hosszu Péter - Krammer Gergely: Papers on Computer Graphics

79/1978 Dr. Adolf Kotzauer:
Beschriftung und Bemassung von automatisch erstellten Zeichnungen unter Benutzung des graphischen dialogs

80/1978 Studies in Applied Stochastic Programming I.

81/1978 Peter Bonitz: Ein Beitrag zur Theorie des Entwurfs doppelt gekrümmter Flächen unter differentialgeometrischen und rechentechnischen Aspekten.

82/1978 Tankó József: Szabályos job-folyam párok ütemezésének vizsgálata I.

83/1978 Tankó József: Szabályos job-folyam párok ütemezésének vizsgálata II.

84/1978 Bányász Csilla - Keviczky László: Discrete Time Identification of Linear Dynamic Process

85/1978 Dr. Hoffmann Péter: Számitógépes szerszámgépvezérlés egy alkatrészprogramozási módszere

86/1978 Ruda Mihály:A SIS77 statisztikai információs rendszer kialakitásának szempontjai, alkalmazásának és továbbfejlesztésének lehetőségei

87/1978 Téli iskola - Operációs rendszerek elmélete

A T A N U L M Á N Y O K sorozatban 1979-ben megjelentek:

88/1979 Renner Gábor - Gaál Balázs - Hermann Gyula - Horváth László - Várady Tamás: Szoborszerü felületek tervezése és megmunkálása

89/1979 Ruda Mihály: A SIS77 statisztikai információs rendszer /a felhasznált számitástechnikai eszközök, a rendszer szerkezete és programjai/

90/1979 Bányász Csilla - Keviczky László: Optimum Insensitivity of the Linear-continuous Transformation

91/1979 Téli iskola /Szentendre/

92/1979 Bolla M., Csáki P., Fischer J., Herodek S., Hoffmann Gy., Kutas T., Telegdi L., Wittman I.: A balatoni ökoszisztéma modellezése

93/1979 Andor László: Kisgépes adatbázis kezelő rendszer

94/1979 Gertler János: Egy statisztikus szürési eljárás számitógépes folyamatirányitásához

95/1979 Báthory M.; Galló V.; Kovács E.; Mérő L.; Siegler A.; Vajta L.: Festőrobot vezérlésére alkalmas alakfelismerési berendezés